Universidad de las Ciencias Informáticas

Centro de Informatización Universitaria

COMPONENTES PARA LA ELABORACIÓN DE DIAGRAMAS DE PRESENTACIÓN EN LA HERRAMIENTA DE DIAGRAMACIÓN DEL PAQUETE ABAD

Facultad 1

TRABAJO DE DIPLOMA

PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS

AUTOR

Adrián Santí Mora

TUTORES

Ing. Enmanuel Azahares Reyes

Ing. Alberto Tamayo Ramos

"La única lucha que se pierde es la que se abandona".

Ernesto Che Guevara

Quiero dedicar este trabajo de diploma, primeramente a mi mamá por haberme dado la vida, por haberme educado, por haber hecho de mí el hombre que soy hoy; a ella se lo debo todo. Quiero dedicar también este trabajo a mi hermanita Liana, ella es la niña linda de la casa, es la niña de mis ojos, y espero que siga mi ejemplo y que también pueda prepararse y estudiar en la universidad. Dedico este trabajo también a mi novia Lisi, por haber creído en mi todo este tiempo, por darme el cariño y el apoyo que necesitaba para poder salir adelante.

Quiero dedicar este trabajo también a toda mi familia, a mis suegros, a los amigos de mi zona que siempre creyeron en mí y estuvieron seguros de que no los iba a defraudar. A mi papá que seguramente debe estar orgulloso de mí. A mis buenos vecinos por apoyarme y a los vecinos envidiosos que gracias a sus comentarios me hice más fuerte. A mis amigos luchadores de la zona (Conrado, Meléndez, Rafa, etc.). A mis amigos luchadores de la escuela (Osdel, Adrián, Asiel, Andy, Guzmán, Sergio). A mis compañeros de apto (Carlos, Dayán, El Luiso, Luisito, Lachy, River, Jose, Mariño, Hilario, Angelito).

Agradecer primeramente a mi mamá por darme la vida, la educación, y la formación necesaria para poder salir adelante en la vida. Agradecer a mí hermanita por su cariño incondicional y por ver en mí un ejemplo a seguir. Agradecer a Lisi por apoyarme durante estos 5 años de carrera, por confiar en mí, por darme el cariño y el amor siempre tan necesarios. Quiero agradecer a mi tocayo Adrián Rivera, por demostrarme que hace rato dejo de ser mi amigo, ahora es mi hermano. A mis tutores, en especial a Yanicet por tirarme esos tremendos cabos con la tesis.

Bueno agradecer también a todos mis amigos, los de mi zona (Chiki, Félix, Mario, Yri, Alieski), a los amigos de la escuela (Carlos, Luiso, River, Dayán, Lachy, Luisito, Angelito, Hilario). A mis amigos luchadores (Osdel, Adrián, Asiel, Andy, Guzmán, Sergio) por entrenar conmigo todas las tardes y de esa forma ayudarme a despejar la mente, cuando las cosas se ponían tensas en esta escuela.

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los _____ días del mes de junio del año 2012.

Adrián Santi Mora

_____

Firma del autor

Ing. Enmanuel Azahares Reyes

_____

Firma del tutor

Ing. Alberto Tamayo Ramos

_____

Firma del tutor

DATOS DE CONTACTO

Ing. Enmanuel Azahares Reyes

El co-tutor Ing. Enmanuel Azahares Reyes. Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2009. Especialista general con dos años de experiencia. Ha trabajado en proyectos productivos en el centro CENIA de la Facultad 1. Se ha desempeñado como desarrollador en el proyecto de Fuerza de Trabajo Calificada, jefe del proyecto Kaínos para la UJC y actualmente es líder del proyecto de Portadores Energéticos.

Correo: eazhares@uci.cu.

Ing. Alberto Tamayo Ramos

Profesor instructor, graduado en el 2007, se desempeña como jefe del Departamento de Universidad Digital del Centro de Informatización Universitaria.

Correo: atamayor.uci.cu.

Resumen

La técnica de diagramación, dentro de la disciplina arquitectura de información cuenta con tres diagramas: presentación, funcionamiento y organización. El presente trabajo de diploma se centra en el diagrama de presentación y su objetivo está orientado a implementar los componentes[1] que se utilizan para realizar dicho diagrama, contribuye al desarrollo de una herramienta que integre los tres tipos de diagramas pertenecientes a la técnica. En el estudio realizado de las herramientas que soportan esta técnica, se identificó que las mismas no realizan el proceso de diagramación, con el nivel de acabado necesario y en muchos de los casos son propietarias, obligando al arquitecto de información a utilizar varias plataformas. El ciclo de desarrollo fue guiado por la metodología de desarrollo SXP y modelado a través del Lenguaje Unificado de Modelado (UML), haciendo uso de la herramienta Visual Paradigm 8.0. Para su implementación el lenguaje de programación Java y como entorno de desarrollo integrado NetBeans 6.9.

Palabras claves: arquitectura de información, técnica de diagramación, diagrama de presentación, prototipo.

[1] Los elementos que forman un vocabulario visual, para realizar los diagramas que describen la arquitectura de información.

Introducción

Disponer de información tiene una importancia fundamental para la sociedad. A lo largo de varios años distintas empresas han reconocido la información como uno de los principales recursos que poseen, siendo la misma un factor crítico para la determinación del éxito o fracaso de sus negocios. Con el fin de satisfacer las crecientes demandas de la sociedad surgen los Sistemas de Información, que garantizan la disponibilidad e integridad de los contenidos.

A mediados de los años noventa aparecen las llamadas nuevas tecnologías de la información y la comunicación, que ofrecieron una nueva forma de la distribución de la información a través de redes globales como Internet. A medida que el desarrollo es mayor, aumenta la cantidad de información disponible en la red y con frecuencia se hace difícil encontrar la información deseada entre tanto contenido. Con el propósito de organizar los contenidos, para ser encontrados y utilizados por los usuarios, de manera simple y directa surge entonces la Arquitectura de Información (AI).

La Arquitectura de la Información debe ser entendida como la disciplina (arte y ciencia) que proporciona métodos y herramientas para estructurar, organizar y representar los contenidos en los sistemas de información. En este sentido, persigue el objetivo de facilitar el acceso a la información contenida en esos entornos y mejorar, así, su utilidad y aprovechamiento por parte de sus usuarios [1].

Las interfaces de usuario que tienen muchos de los productos informáticos en la actualidad no están enfocadas al modelo mental de los usuarios finales. Esto ha provocado demora en las búsquedas de información restándole utilidad a los sistemas informáticos. La Universidad de las Ciencias Informáticas (UCI) pretende incorporar a sus procesos de desarrollo de *software* disciplinas como la AI. Con el objetivo conseguir mejores resultados de organización y disposición de los contenidos en los productos. Para lograr que un usuario al entrar por primera vez al sitio pueda encontrar fácilmente la información deseada; lo pueda entender en forma rápida y sin esfuerzo.

Con el objetivo de garantizar que los proyectos de la UCI incorporen estas prácticas en el desarrollo de sus productos, se creó en el Centro de Informatización Universitaria (CENIA), el proyecto paquete de herramientas para diseñar experiencia de usuario (Abad). En el que se desarrolla una solución integrada

para la gestión de los flujos de AI en los proyectos de desarrollo de soluciones informáticas de la universidad.

Como servicio el proyecto Abad brinda una gestión integrada de los artefactos resultados de las actividades de experiencias de usuario. Una de estas actividades es la técnica de diagramación que consiste en la representación de los contenidos que tendrá un producto digital, y se encarga de la organización y representación de los mismos. Esta técnica para su desarrollo utiliza tres tipos de diagramas que a efectos de este trabajo de diploma se clasifican en: diagramas de organización, de funcionamiento y de presentación. Hacia este último diagrama, se centra el objetivo de este trabajo. Los diagramas de presentación o prototipos de interfaces de usuario como también se les conoce, son términos que se utilizan indistintamente en este trabajo de diploma. Pero se refieren al mismo proceso, de organizar y estructurar los contenidos en las interfaces gráficas de los productos informáticos.

En la actualidad existen diferentes aplicaciones que son utilizados por los arquitectos de información, para la elaboración de los diferentes tipos de diagramas, algunas muy conocidas como el Microsoft Visio y el Axure. Actualmente no existe una herramienta que integre la confección de los tres diagramas, obligando al arquitecto de información a interactuar con varias herramientas al mismo tiempo y en muchos de los casos con más de un sistema operativo. Lo que hace este proceso más lento y engorroso. Es por ese motivo que el proyecto Abad realiza una herramienta que pretende integrar los tres tipos de diagramas mencionados anteriormente. Esta herramienta no cuenta con los componentes de presentación para realizar un prototipo de interfaz de usuario.

Dada la situación antes expuesta surge el siguiente **problema de la investigación:** ¿Cómo facilitarle al arquitecto de información la representación de los contenidos en el prototipado de interfaces de usuarios?

Según el problema identificado anteriormente el **objeto de estudio** se enmarca en: el proceso de arquitectura de información, y se define como **campo de acción** en: la técnica de diagramación dentro de la arquitectura de información.

Para dar solución al problema planteado se traza como **objetivo general**: implementar los componentes de presentación en la herramienta de diagramación para facilitarle al arquitecto de información, la realización del prototipado de interfaz de usuario. Definiendo como **objetivos específicos:**

- Identificar los principios teóricos de la investigación relacionados con la técnica de diagramación dentro de la arquitectura de información.
- Desarrollar la propuesta a partir del análisis realizado de los componentes de presentación para la herramienta de diagramación.
- Realizar las pruebas a los componentes de presentación de la herramienta de diagramación.

Se utilizaron como **métodos de investigación científica**:

Métodos Teóricos

Analítico-sintético: a través de este método se pudo establecer conceptos, expectativas y necesidades de la disciplina AI y en particular de la técnica de diagramación. Mediante la revisión bibliográfica lo que permitió el análisis de todos los elementos que utiliza el arquitecto de información en la modelación de los productos, esta acción posibilitó identificar y sintetizar las necesidades y características que presentan.

Histórico-lógico: se utilizó para el estudio de la trayectoria histórica de la disciplina de AI y en especial la diagramación. Lo que permitió elaborar una sucesión cronológica y argumentada que facilitó el estudio del objeto en el presente trabajo de diploma. Apoyándose en la lógica de todo el proceso que permite definir la línea de investigación según los antecedentes, características y necesidades, lo que posibilitó definir las carencias y dificultades de los productos analizados que motivan la realización de esta aplicación.

El presente documento consta de 3 capítulos, estructurados de la siguiente forma:

Capítulo 1: Fundamentación Teórica.

En este capítulo se analizan los principales conceptos relacionados con la investigación que serán utilizados en el desarrollo del trabajo de diploma. Se realiza un estudio de las soluciones homólogas existentes. Se definen las herramientas, tecnologías y metodología a usar para la construcción del sistema.

Capítulo 2: Construcción del Sistema.

Se define la arquitectura y los patrones de diseño a utilizar. Se realizan los modelos necesarios para el desarrollo de los componentes. Se implementan los componentes.

Capítulo 3: Pruebas del Sistema.

Se diseñan y aplican las pruebas al sistema para comprobar el correcto funcionamiento de las principales funcionalidades de la aplicación.

Capítulo 1: Fundamentación teórica

1.1. Introducción

En este capítulo se exponen los aspectos generales de la AI así como los de la herramienta de prototipado, específicamente los prototipos de interfaz de usuario que a efectos de este trabajo se refiere a los diagramas de presentación dentro de la AI. Se analizan sistemas que realizan funciones similares a las que se presentan en la solución propuesta. Además, se realiza un análisis de la metodología utilizada por el analista y el lenguaje de programación empleado en el desarrollo del sistema informático.

1.2. Conceptos fundamentales

1.2.1 Arquitectura de Información

La arquitectura de la información es la disciplina encargada del estudio, análisis, organización, disposición y estructuración de los contenidos en espacios de información. En la actualidad es muy utilizada en el diseño de las interfaces para sitios web. Estudia cómo la organización de los contenidos puede ayudar a definir el pensamiento de las personas dada la adaptabilidad que tienen estos procesos en varios aspectos de la vida, ya que pueden ser usados en ramas de la medicina, tales como la psicología y la psiquiatría. Es una materia que propone al usuario como vía fundamental para la solución de sus propios problemas, revelando sus criterios sin necesidad de que este los exprese mediante la palabra, de manera que se logre una comunicación estándar entre ambas partes [2].

Esta disciplina trata de organizar la información, de manera que los usuarios puedan encontrar las respuestas correctas a sus interrogantes. Para esto se vale de varias técnicas entre las que se destacan los métodos de diseño centrados en el usuario y dentro de estos se encuentran el ordenamiento de tarjetas y el análisis de secuencia [2].

1.2.2 Diagramación

La diagramación dentro de la arquitectura de la información: es la organización de los contenidos del producto, su funcionamiento básico, y la ubicación que tendrán estos en la interfaz gráfica. Un grupo de autores, como *Dan Brown, Jesse James Garret,* pioneros en los temas del diseño y representación de los

productos informáticos, dividen estos diagramas en 2 tipos [3].

Planos (Como se conoce en el idioma Inglés *Blueprints*).

Maquetas (Denominada así en el idioma Inglés *Wireframes*).

El primer tipo de diagrama (plano) tiene como objetivo representar las principales áreas de organización y rotulado, están enfocados a los aspectos estructurales y de funcionamiento del producto. Generalmente se representan con textos, cajas y flechas. Su función es reflejar y facilitar la toma de decisiones en cuanto a diseño, manteniendo la comunicación constante de dichos cambios al resto del equipo de desarrollo.

Christina Wodtke conceptualiza los *Blueprint* como: "Un plano de diseño es justamente una buena idea llevada a la realidad a través de la escritura".

El segundo grupo (maqueta) tienen el objetivo de mostrar el contenido de las páginas, utilizando los elementos que se definieron en los planos y ubicándolos en las páginas o pantallas del producto final. Están comprendidos como prototipos de baja fidelidad, pues se realizan a escala de grises y no muestran el diseño gráfico del producto ni la funcionalidad de sus códigos de programación [3].

Los niveles asociados al prototipado son:

- Prototipos de baja fidelidad o estáticos.
- Prototipos de fidelidad intermedia (diseño gráfico).
- Prototipos de alta fidelidad o dinámicos (Web, HTML).

Un estudio del Lic. Rodrigo Ronda León, propone un sistema de diagramación donde son creados tres tipos de diagramas teniendo en cuenta las funciones que debe cumplir el arquitecto de información durante el diseño de un producto digital [4]. Estos diagramas son:

- Diagramas de organización (planos).

- Diagramas de funcionamiento (planos avanzados).
- Diagramas de presentación (presentación).

Está clasificación, no significa que un diagrama pueda sustituir a otro, sino todo lo contrario, deben estar relacionados entre ellos, para que cada diagrama complemente el anterior y sirva de referencia al siguiente. Esta categorización de los diagramas no significa que deben existir siempre los tres tipos diagramas [4].

Después de realizar un estudio sobre las diferentes formas de ejecutar la técnica de diagramación se ha decidido utilizar el sistema de diagramación del autor Rodrigo Ronda León que está basado en la creación de tres tipos de diagramas (organización, funcionamiento, presentación) ya que la mayoría de los arquitectos de información en la UCI para lo que en un principio se le orienta esta herramienta, utilizan un esquema estrechamente relacionado con los tres tipos expuesto anteriormente [4].

1.2.3 Prototipado

Desde que se empieza el desarrollo de un sistema interactivo se necesita probar partes del mismo con multitud de objetivos para: verificar funcionalidades, averiguar aspectos relacionados con la interfaz del sistema (posición de controles, textos, colores), validar la navegación, probar nuevas posibilidades técnicas, etc. Es impensable llegar al final del desarrollo sin haber realizado comprobaciones a lo largo del camino. Los prototipos de interfaz de usuario, constituyen el mecanismo que permite realizar estas comprobaciones. Los prototipos son documentos, diseños o sistemas que simulan o tienen implementadas partes del sistema final, constituyen una herramienta muy útil para, hacer participar al usuario en el desarrollo y poder evaluar el producto desde las primeras fases del desarrollo [5].

En las aproximaciones de los Diagramas de Casos de Usos (DCU), los prototipos constituyen mucho más que simples demostraciones del producto; se utilizan para recoger las impresiones del usuario para repercutirlas en el diseño de la interfaz. Un prototipo en sentido genérico es una implementación parcial pero concreta de un sistema o una parte del mismo que principalmente se crea para explorar cuestiones sobre aspectos muy diversos del sistema durante su desarrollo. En referencia a una interfaz de usuario se

realizan prototipos con la finalidad de explorar los aspectos interactivos del sistema incluyendo la usabilidad, la accesibilidad y/o la funcionalidad del mismo [6].

El uso de los prototipos en el desarrollo de sistemas informáticos no se limita solamente a probar las interacciones que los usuarios deben realizar, sino que son útiles también para otras actividades que se realizan durante el proceso, como por ejemplo su gran utilidad en la fase de recogida o análisis de requisitos, lo que amplía y mejora la información necesaria para el desarrollo del sistema.

Características de los prototipos:

- Son herramientas muy útiles para fomentar la comunicación entre todos los componentes del equipo de desarrollo y los usuarios.
- También incitan a que los usuarios se integren activamente en el desarrollo.
- Ayuda a los diseñadores a escoger cuando tienen varias alternativas de diseño.
- Permiten evaluar el sistema desde las primeras fases del desarrollo.
- Son de gran ayuda para la documentación del proyecto.
- Es una herramienta muy útil para ver implementadas ideas abstractas y que se conviertan en visibles y *probables*.
- Mejoran la calidad y completitud de las especificaciones.

Las diferentes técnicas de realización de prototipos varían por la fidelidad de estos respecto al sistema final. Será necesario, por tanto, valorar en cada momento cuál es la técnica más apropiada a utilizar en función del período del desarrollo en el que se encuentra y de los objetivos a cumplir. Estos suelen catalogarse en dos categorías básicas: baja fidelidad y alta fidelidad. Los prototipos de baja fidelidad implementan aspectos generales del sistema sin entrar en detalles. Permiten abarcar un espectro mayor de la interacción a realizar [5].

Con los prototipos de alta fidelidad se representa aspectos más precisos. Sirven por ejemplo, para detallar el proceso interactivo global de una o varias tareas concretas. Los prototipos de baja fidelidad se caracterizan por ser económicos, rápidos de construir, rápidos de arreglar y no precisan de técnicos

expertos. Mientras que los prototipos de alta fidelidad, se caracterizan por el uso de herramientas especializadas de prototipado, que ofrecen más detalle y precisión por requerir de expertos que conozcan dichas herramientas, por ser más caros y por necesitar más tiempo para realizar dichos prototipos [6].

Cada una de estas categorías básicas posee sus características específicas. Ciertas pruebas, por ejemplo de rendimiento, solo pueden realizarse o se obtienen mejores resultados mediante prototipos de alta fidelidad. También depende de la fase del desarrollo en que se encuentren la aplicación serán más útiles unos u otros.

Aquí se resume las ventajas y los inconvenientes de las dos categorías de técnicas de prototipado mencionadas:

Prototipos de baja fidelidad

- **Ventajas**
  - Costes de desarrollo pequeño.
  - De muy rápida creación.
  - Fácil de cambiar (cualquiera puede realizar los cambios).
  - Los usuarios, al ser conscientes de la facilidad de los cambios del bajo coste económico, se sienten cómodos para opinar y proponer cambios.
  - Evaluación de múltiples conceptos de diseño.
  - Útil para el diseño general de las interfaces.
  - Útil para identificar requisitos.
  - Sensación de prueba.
- **Inconvenientes**
  - Limitado para la corrección de errores.
  - Especificaciones poco detalladas (para pasar a la codificación).
  - Dirigido por el evaluador.

- Su utilidad disminuye cuando los requisitos ya están bien establecidos.
- Navegación y flujo de acciones limitadas.

Prototipos de alta fidelidad

- **Ventajas**
  - Funcionalidad de tareas completa.
  - Completamente interactivo.
  - Dirigido por el usuario.
  - Navegabilidad
  - Aspecto semejante al sistema final.
  - Puede servir como especificación.
  - Puede servir como herramienta promocional y para demostraciones de ventas.
- **Inconvenientes**
  - Elevados costes de desarrollo.
  - Requieren mucho tiempo de implementación.
  - Mayor dificultad de cambiar (cambios solo realizables por el autor y requieren mayor tiempo).
  - Crea falsas expectaciones.
  - Menor efectividad para la recolección de requisitos.

Con el desarrollo de este trabajo de diploma se realizará la implementación de los componentes de presentación, pertenecientes a la herramienta de diagramación del paquete Abad. Herramienta que permitirá al arquitecto de información la realización de prototipos de interfaz de usuario, con un nivel básico de seguridad.

1.3. Estudio de herramientas existentes

En este apartado se realiza un estudio de una serie de herramientas, las cuales son utilizadas para la creación de prototipos de interfaz de usuario. De las que se analizan sus características y funcionalidades con el objetivo de incorporarlas a la solución de este trabajo de diploma.

1.3.1. Justinmimn

Permite crear interfaces para aplicaciones de escritorio, web y móviles. Prototipa y simula aplicaciones web 2.0, portales, intranets y sitios webs, aplicaciones para móviles (iPhone, BlackBerry) y aplicaciones para escritorio. Opera bajo los sistemas operativos Windows 2000, XP, Vista, 7 y Mac v10.4 Tiger V10.5 y Leopard [7].

Es completamente funcional, todos los elementos que se pueden utilizar en los prototipos en el momento de la simulación tienen el comportamiento esperado, *checkbox*, listas y enlaces a otras páginas. Genera toda la documentación necesaria del proyecto, tanto del prototipo en sí mismo como los requisitos.

El programa es muy fácil de utilizar y muy intuitivo, los menús están bien organizados y son muy visibles. En un principio en la versión de prueba gratuita los componentes quizás son un poco escasos, pero suficientes para realizar un prototipo con las funcionalidades normales. Para la creación del prototipo únicamente arrastrando los elementos en la zona central comienza a crearlos, además contiene manuales y videos de ayuda para la creación del primer prototipo. También con soporte vía web. La creación del prototipo se realiza rápidamente y con pocos fallos en el momento de la implementación. Una característica importante que posee esta herramienta, por la cual no pudo ser seleccionada, es que es una herramienta propietaria [7].

1.3.2. Lairbuilder

Permite simular páginas web o aplicaciones de escritorio sobre AIR (Adobe AIR es una tecnología que permite la creación de aplicaciones de escritorio a partir de tecnologías de desarrollo de páginas web, como pueden ser HTML (lenguaje de marcado de hipertexto), Ajax o Flash) y aplicaciones para iPhone [8].

Los archivos los guarda en formato FBP (estilo constructor de proyecto) se tiene que tener instalado el programa que facilita el propio fabricante para poder realizar la evaluación de los prototipos creados [8]. Tiene un elevado grado de funcionalidad, los elementos que requieren un mínimo de interacción la suelen completar además permite el control por ejemplo de contenido multimedia completamente funcional [8].

Contiene el menú de funcionalidades a la izquierda sin ningún tipo de filtro con la navegación por pestañas. En estas pestañas se encuentra el primer bloque que contiene los elementos y las librerías. El segundo bloque que contiene las propiedades del elemento seleccionado y los eventos del mismo y el último bloque que contiene el sistema de navegación del prototipo. Para encontrar cualquier otra funcionalidad que no esté en este menú situado a la izquierda la tarea se vuelve complicada y pesada [9].

Está más enfocada para la creación rápida de un prototipo, no a las funcionalidades que debe brindar una herramienta de este tipo. No genera ningún tipo de documentación respecto al proyecto y únicamente deja anotar observaciones en los elementos del mismo. Opera solo sobre sistema operativo Windows, no presenta una interfaz visual intuitiva que permita interactuar fácilmente con la herramienta. No permite llevar un control de los requisitos del prototipo como lo hacen otras herramientas o el diagrama de escenarios [8].

1.3.3. Axure

Únicamente permite crear prototipos y simulaciones para páginas web en HTML (lenguaje de marcado de hipertexto) y aplicaciones de escritorio, aunque claramente está creado para realizar prototipos para páginas web. No crea prototipos para ningún otro tipo de plataforma. Soporta simulaciones de una fidelidad media. Funciona bajo los sistemas operativos Windows 2000, XP, 2003 Server, 2007 [5].

Los prototipos realizados se pueden evaluar a través de los navegadores web más habituales, reporta los prototipos en HTML (lenguaje de marcado de hipertexto) en forma de árbol y con todos los comentarios o anotaciones que se pueden poner en los elementos. Los prototipos, que se van organizando a modo de árbol, son exportados a HTML (lenguaje de marcado de hipertexto) y los comentarios de cada elemento se pueden ir viendo sobre la propia presentación. Este sistema facilita muchísimo la presentación de los prototipos manteniendo anotados todos los comentarios sobre cada elemento.

Dichos comentarios también se pueden exportar en formato Microsoft Word, lo cual facilita su tratamiento o posterior inclusión en la documentación final. Todos los elementos que se añadan a un prototipo son completamente funcionales, además permite incluir información adicional a cada elemento, por ejemplo la definición de estado (propuesto, aprobado, incorporado) o el beneficio que aporta (crítico, importante, útil). En el prototipo quedan especificadas todas las características de los elementos que se disponen y en la simulación aparecen tal cual han quedado definidos [5].

Es muy fácil de utilizar, la estructura de la aplicación sigue la línea de cualquier aplicación de Windows, un menú de opciones en la parte superior y un menú lateral izquierdo con todas las posibilidades de diseño. Para empezar a crear el prototipo se selecciona el elemento deseado y es arrastrado a la zona central. En el lateral derecho, en el momento de la creación de un elemento, se editan todas sus características tanto de funcionalidad como meras anotaciones importantes en el momento de especificar los requisitos o la documentación.

Permite componer la página web visualmente, añadiendo, quitando y modificando los elementos con suma facilidad. Dispone de una serie de objetos predefinidos que cubren la mayoría de las necesidades que puede tener un desarrollador la hora de integrar elementos en un prototipo. Solo es necesario arrastrar y soltar sobre una rejilla [5], por último, mencionar que permite la creación de plantillas, es decir, conjuntos de objetos resultados de la combinación de otros. Es fácil aprender el funcionamiento de los prototipos y los requisitos de los objetos con esta herramienta, ya que es muy sencillo crearlos y concretar todas las especificaciones [10].

1.3.4. ForeUI

Permite simular páginas web y aplicaciones de escritorio para los sistemas operativos más comunes, versiones de Windows, Mac, Linux y Solaris, es propietaria y permite simular aplicaciones para móviles como iPhone [5]. Los prototipos realizados pueden ser visualizados fácilmente en cualquier explorador web. Posee dos opciones de visualización, como una presentación que posee la facilidad de tener un puntero donde remarcar elementos de prototipo o hacer señalizaciones sobre él, y a través de cualquier navegador web que soporte Java donde el prototipo es completamente funcional. También se puede exportar el prototipo en HTML (lenguaje de marcado de hipertexto) o PDF (formato de documento portátil) [11].

Todos los elementos que se pueden editar en el prototipo son completamente funcionales. En el modo de visualización se puede interactuar con todos estos elementos de forma "real" sin ningún tipo de restricción, salvo las que sean detalladas específicamente en cada elemento. Un aspecto interesante que tiene esta herramienta es que permite especificar el aspecto que debe tener el prototipo según el sistema operativo en el que va a estar alojado e incluso se puede realizar a modo guion gráfico [11].

La aplicación es fácil de utilizar, como es habitual en este tipo de herramientas la edición del prototipo se realiza con el arrastre de los elementos que se desean introducir, tras esta acción se abren todas las ventanas relacionadas con este elemento para poderlo editar. Las opciones y menús están muy bien organizados. Se limita a evaluar el modo en el que se puede crear prototipos completos y su visualización para poderlos evaluar. No genera ningún tipo de documentación relativa al proyecto del prototipo en general y es un poco escasa en cuanto a la especificación de los requisitos de los elementos. La única especificación que se puede realizar de los elementos es el tipo del elemento [11].

1.3.5. Mockup Screens

Permite crear simulaciones de aplicaciones de escritorio para sistemas operativos *Windows* y diseño de *blogs* o páginas web. No permite la creación de prototipos de aplicaciones para móviles u otros tipos de dispositivos portátiles. Los prototipos que se crean con esta herramienta, no se pueden visualizar para su evaluación como prototipo, únicamente se puede visualizar como en capturas de pantalla donde se puede observar las anotaciones que se hayan podido insertar en cada elemento. También se puede observar como crea una documentación con las capturas de pantalla del proyecto y sus anotaciones, pero nada de especificaciones de requerimientos, ni ninguna otra información extra que se podría aportar del prototipo [5].

Aunque en las tablas de datos se incluyen los datos reales, incluso copiar o importar una tabla de Microsoft Excel. El hecho de no poder probar para su evaluación el prototipo deja mucho que desear en cuanto a funcionalidad se refiere. Es una herramienta muy fácil de utilizar, la interfaz es muy intuitiva y la información en ella está bien organizada. Como en el resto de las aplicaciones de este tipo, para crear el prototipo se utilizará la técnica de arrastrar al centro de la pantalla. La facilidad de uso de esta herramienta es proporcional a la simplicidad de la misma, puesto que no ofrece muchas funcionalidades resulta fácil familiarizarse con ella [5].

Como se ha podido ver en casos anteriores, esta es una herramienta más para realizar capturas de pantalla de las futuras aplicaciones, pero con poca funcionalidad extra. En cuanto a la documentación, genera un tipo de documentación encarado a las presentaciones empresariales del producto, pero no al estudio del prototipo. Como aspecto negativo mencionar que no permite llevar ningún tipo de control sobre los requisitos de los elementos [5].

1.3.6. Mockflow

Herramienta bastante completa que permite simular páginas web y aplicaciones de escritorio para computadoras y aplicaciones para móvil. Aunque es una herramienta muy completa y se pueden ver los prototipos con el aspecto que van a tener en la plataforma para la cual se realizan, no es posible visualizarlos para que sean evaluados, únicamente pueden ser visualizados en la pantalla sin ningún tipo de funcionalidad o exportados en PDF (formato de documento portátil), PowerPoint o imágenes [12].

Es una herramienta de las más completas en cuanto a elementos disponibles para poder realizar un prototipo. También se puede compartir con la comunidad de la misma aplicación y utilizar *plantillas* creados por otras personas [12]. Mediante la técnica de arrastre de los elementos se puede empezar a crear el prototipo. Los elementos no están organizados en el menú, pero este es muy intuitivo y muy visual lo que hace que la información no parezca desordenada, y se puede encontrar fácilmente. Tiene una estética muy atractiva y a la vez práctica para la navegación por su menú en la derecha y el menú de navegación superior.

La herramienta posee un gran número de elementos de diseño, pero no aporta mucha información en cuanto al aspecto de aprendizaje del método de la creación de prototipos. Con esta aplicación únicamente se puede ver el sistema de la futura aplicación o página web. No permite especificar requisitos ni añadir características o restricciones a los elementos [12].

1.3.7. ExtremePlanner

Permite simular aplicaciones de escritorio para Windows, Mac o Linux y páginas web. No se puede utilizar en aplicaciones para recursos móviles u otras plataformas similares. Se pueden visualizar los prototipos creados a través de cualquier navegador web, en los sistemas operativos especificados anteriormente. En

la visualización se puede ver tanto una descripción de la página que se visita actualmente, un listado con las etiquetas de los elementos utilizados y el listado de páginas del proyecto. Para visualizarlo se presiona el botón correspondiente a la vista previa y permite ver el proyecto completamente funcional [13].

La aplicación viene con tres páginas de ejemplo de autenticación de usuarios. Estas páginas son completamente funcionales, pero en el momento de crear nuevas páginas con nuevos elementos es realmente difícil debido a que no vienen elementos predefinidos. Tienen que ser creados con una herramienta de dibujo y definirles etiquetas de comportamiento y especificaciones. Es realmente difícil de utilizar porque todos los elementos que se quieran incluir en el prototipo, deben ser creados desde cero, debido a que no existen elementos predefinidos. La creación de los elementos desde cero no es trivial, pues la aplicación no resulta fácil de utilizar para las pocas prestaciones que posee.

Al usar esta herramienta se pierde más tiempo componiendo elementos, que con otras aplicaciones que ya vienen predefinidas prestando atención al proceso de creación del prototipo. La creación resulta engorrosa y lenta. En la documentación que genera no detalla la especificación de requisitos ni los diagramas de navegación del sitio, únicamente muestra los elementos que componen cada página (de manera generalizada) y la función que tienen [13].

1.3.8. Pidoco

Esta aplicación únicamente permite simular aplicaciones de escritorio para Windows y páginas web. Para evaluar los prototipos se selecciona el botón correspondiente y el navegador web que esté instalado en la computadora se abre. La visualización del prototipo es completamente funcional y se pueden seleccionar dos tipos de vistas. La manera real donde pueden verse todas las características de las páginas como en el caso real, y la visualización plana como si fuera de un prototipo a papel, que se ve con trazos a mano, en blanco y negro pero igualmente funcional. Esta herramienta también permite realizar las pruebas de usabilidad sin necesidad de descargar u obtener licencia de ninguna aplicación [5].

Una vez colocado en el escenario se asignan numerosas propiedades de las galerías de símbolos como la vinculación a otros elementos. El uso de este método permite crear proyectos plenamente interactivos o compartir mediante prototipos y presentar conceptos de diseño visual a compañeros de proyecto. Cada página consta de diferentes capas, que se pueden ver como interfaz en bloques reutilizables de

construcción, hacen que trabajar con esta herramienta sea más sencillo y eficiente. Los elementos que aparecen en varias páginas de los prototipos (tales como barras de navegación, gráficos o logos) solo tienen que ser definidos una vez, posteriormente si existe alguna modificación al actualizar una página se aplica al resto de páginas donde aparecen [14].

Esta herramienta es realmente fácil de usar, trabajar con ella es ágil y eficiente. Aunque no tiene un aspecto parecido a Windows la navegación por ella es agradable. Dispone de un menú superior de navegación y del menú de la izquierda con los elementos. Aun así hay funciones específicas que son un poco más complicadas de encontrar. Es una herramienta encarada al trabajo empresarial de la creación de prototipos ya que es rápida y muy colaborativa, pero pasa por alto algunos elementos que son importantes para el aprendizaje del proceso de creación, como por ejemplo la documentación [14].

1.3.9. Microsoft Visio

Permite simular aplicaciones de escritorio y web de forma generalizada. La función principal de este programa no es la creación de prototipos, pero se tuvo en cuenta para realizar el análisis, porque es una herramienta muy utilizada en la realización de los prototipos. El mismo programa Microsoft Visio puede realizar la visualización del prototipo pero únicamente como una captura de pantalla, ninguno de los elementos que puede tener el prototipo es funcional por lo que resulta difícil imaginar cómo quedará el proyecto. En este aspecto Visio no ofrece buena visualización. La opción que brinda para visualizar está situada en el menú de opciones de la parte superior, llamada vista previa. Esta herramienta tampoco incorpora ninguna opción para poder compartir el proyecto o poderlo evaluar al mismo tiempo con compañeros de proyecto [5].

Los pocos elementos que contiene la aplicación para poder crear los prototipos no son funcionales. Únicamente se puede especificar su nombre o modificar un poco su aspecto, pero la mayoría no tienen el comportamiento esperado, característica normal si se tiene presente que la visualización de los proyectos no es funcional [5]. Debido a que esta herramienta es de dibujo vectorial y no está creada específicamente para crear prototipos no es muy fácil de utilizar.

Utiliza el mismo sistema de arrastre a la zona central para crear los prototipos, dispone de pocos elementos predefinidos, no obstante, dispone de plantillas que pueden resultar de mucha ayuda para

crear los prototipos. Al ser una herramienta de Microsoft está a mano de los usuarios que puedan comprarla. Su navegación resulta fácil, debido a la familiarización con ellas, pero posee muchas prestaciones que no se utilizan para crear prototipos y que pueden llegar a causar molestias. Como se ha mencionado, no es una herramienta creada para este fin exclusivo, por lo que se considera que no es apta para el proceso de creación de prototipos [5].

1.3.10. Pencil Project

Es una solución informática en lengua inglesa, gratuita y de código abierto disponible para ordenadores con sistema operativo Windows o GNU/Linux. Es una herramienta desarrolla en el lenguaje XUL (lenguaje de interfaz usuario) razón por la que no se puede utilizar esta herramienta en la solución de este trabajo. También puede ser instalada como una extensión en el navegador Firefox.

Es una solución muy versátil que ofrece una librería formada por más de cincuenta elementos gráficos para el prototipado web, que admiten una edición posterior y la incorporación de nuevos elementos gráficos externos. Permite la exportación a diferentes formatos como HTML (lenguaje de modelado de hipertexto), Png (gráficos de red portátil), *Open Office*, *Microsoft Word*, PDF (formato de documento portátil); pero sólo interesa su propio formato. Permite la creación de prototipos dinámicos y anotaciones; pero no las notas a pie de página ni la edición de proyectos de creación colaborativa de prototipado [15].

1.3.11 Conclusiones de estudio

En el estudio realizado anteriormente se identificaron y analizaron una serie de herramientas que son utilizadas para la realización de prototipos de interfaz de usuario. Pero la mayoría de estas no permite la integración de los tres tipos de diagramas pertenecientes a la técnica de diagramación (organización, flujo, presentación), y las que lo hacen, es a un nivel muy básico, lo que impide que sea escogida una de ellas para la solución de este trabajo. En muchos de los casos estas herramientas son de carácter propietario, lo que impide que sean utilizadas en la universidad, debido a la política de migración a *software* libre. Otros de los aspectos negativos identificados para la selección de una de estas herramientas, es que muchas están implementadas en un lenguaje de programación que no es compatible con el que se está desarrollando la herramienta de diagramación (Java) y en otros casos no son multiplataforma.

De igual forma en este estudio se identificaron una serie de características, que son de utilidad para la obtención de un producto con la calidad requerida. Se identificaron una serie de componentes de presentación, que son de gran utilidad, y permiten realizar un prototipo con mayores funcionalidades. Otras de las características que se debe incorporar a la solución del producto, es la manera en que se pueden editar los componentes de presentación, una vez que se encuentran en el área de trabajo. A estos se le gestionan sus propiedades mediante una paleta de propiedades. Como es el caso del ForeUI, que presenta una paleta flotante que permite editar las propiedades y los eventos de los componentes. También se identificó la funcionalidad que permite exportar como página HTML (lenguaje de marcado de hipertexto) un prototipo realizado.

1.4. Tecnologías

Para la realización de una aplicación se deben tener en cuenta las tendencias actuales en torno a las herramientas a utilizar, para el desarrollo de la misma, de forma que el trabajo se haga más sencillo y a la vez con mayor calidad. Se deben utilizar tecnologías avanzadas y en continuo progreso, para que de esta forma la aplicación cumpla con las expectativas de los usuarios finales y además se pueda actualizar la misma con gran facilidad.

En el mundo de las comunicaciones uno de los temas más investigados actualmente es el de las metodologías de desarrollo de *software*. Estas en su mayoría establecen cómo trabajar eficientemente buscando evitar el fracaso de los proyectos. Tienen como objetivo aumentar la calidad del sistema que se produce en todas sus fases de desarrollo, define un conjunto de procesos y actividades que guían a los desarrolladores durante el ciclo de vida de un proyecto, especifican los artefactos a construir y organizan el personal involucrado en roles dentro del equipo de trabajo.

Las metodologías y tecnologías que se describen en este capítulo, son conocidas como metodologías ágiles de desarrollo de *software*. Las que serán utilizadas para el desarrollo de esta herramienta. Las mismas son las que propone el centro de informatización universitaria (CENIA), las cuales son utilizadas por el proyecto paquete de herramientas para diseñar experiencia de usuario (Abad).

1.4.1. Metodologías ágiles de desarrollo de software

Estas metodologías se diferencian por su sencillez, rápido entendimiento y poca documentación. Son metodologías centradas en la necesidad del cliente, siendo necesaria la presencia de este, en todo momento, durante todo el ciclo de vida del proyecto. Son especialmente útiles en proyectos donde es necesario hacer cambios en el desarrollo constantemente. Dentro de las metodologías ágiles se destacan el Scrum y el XP (programación extrema), estas pueden ser utilizadas de forma independiente o como la combinación de ambas, formando el Scrum-XP.

Algunas características de las metodologías ágiles:

- Un desarrollo iterativo, ágil, dinámico y muy flexible.
- Ubica el programa que funciona por encima de la documentación exhaustiva.
- Prioriza la colaboración con el cliente, por encima de la negociación contractual.
- Sitúa la respuesta al cambio, por encima del seguimiento de un plan.
- Propone a los individuos y su interacción, por encima de los procesos y las herramientas.

1.4.1.1. Programación extrema (XP)

Programación extrema (XP) surgió a finales de los 90s, cuando la industria del *software* estaba bajo tres grandes influencias, el cambio de paradigma de programación estructurada a programación orientada a objetos, el auge de internet y el crecimiento de las organizaciones como factores de competitividad para los negocios[16]. La programación extrema brinda especial atención al trabajo en equipo. Los líderes, clientes y desarrolladores son socios iguales en un conjunto de colaboración. Implementa un entorno sencillo, pero eficaz que permite alcanzar una alta productividad.

El equipo de desarrollo está facultado para tomar decisiones en torno al problema para solucionarlo lo más eficientemente posible. El objetivo principal de XP es reducir el costo del cambio, usando principios, valores y prácticas básicas, por medio de las cuales un proyecto debe ser más flexible al cambio. Otros objetivos son reconciliar humanidad y productividad, establecer un mecanismo de cambio social y establecer una disciplina de desarrollo de *software* [16].

1.4.1.2. Scrum

Metodología basada en un proceso iterativo e incremental, en la que se aplican de manera regular un conjunto de buenas prácticas para trabajar en equipo. Está especialmente indicada para proyectos en entornos complejos, en los que se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos y la innovación, la competitividad, la flexibilidad y la productividad son fundamentales. El equipo de desarrollo tiene la completa responsabilidad de decidir la mejor manera de trabajar para ser lo más productivo posible.

Scrum permite la creación de equipos motivados, capaces de organizarse por sí mismos, donde la comunicación y la transparencia son totales. Con esta metodología, el usuario gana protagonismo y el cliente se convierte en parte del equipo de desarrollo. Permite además seguir de forma clara el avance de las tareas a realizar, de manera que los líderes puedan ver día a día cómo progresa el trabajo. Sin embargo, Scrum más que una metodología de desarrollo de software, es una forma de auto-gestión de los equipos de programadores. Debería, por tanto, complementarse con alguna otra metodología de desarrollo [16]. Scrum más que una metodología de desarrollo de software, es una forma de auto-gestión de los equipos de programadores. Debería, por tanto, complementarse con alguna otra metodología de desarrollo.

1.4.1.3. Scrum-XP

SXP es un híbrido de la unión de Scrum y XP que ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de *software* para el mejoramiento de la actividad productiva. Fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo y ayudando al líder del proyecto a tener un mejor control del mismo. Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. Está basado completamente en los valores y principios de las metodologías ágiles. Como método de estimación se utiliza la opinión de expertos y consta con métricas o indicadores para lograr una eficiente calidad [16].

Consta de 4 fases principales: la fase de Planificación-Definición, en donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto. La fase de Desarrollo, en donde se realiza la implementación del sistema hasta que esté listo para ser entregado, la fase de

Entrega, en donde se pone en marcha el producto y por último la fase de Mantenimiento, donde se realiza el soporte para el cliente. De cada una de ellas se despliegan 7 flujos de trabajo: concepción inicial, captura de requisitos, diseño con metáforas, implantación, prueba, entrega de la documentación y soporte e investigación. El cual se utiliza por el equipo de desarrollo cuando sea necesario, es decir, es un flujo que se puede mover y utilizar en cualquier parte del ciclo de vida del proyecto [17].

SXP plantea una organización de la documentación de cada uno de los sistemas de forma eficiente, además de que cada uno de los miembros del equipo se muestre interesado y motivado para el desarrollo. El trabajo se agiliza y mantiene al cliente dentro del equipo de desarrollo, proporcionando mejores resultados y una mayor satisfacción por parte de los interesados finales del producto [17].

En la realización de este trabajo de diploma, concepción de la herramienta de diagramación se decide utilizar esta metodología por todas las ventajas que trae en un desarrollo de este tipo. Este trabajo parte de analizar y refinar los artefactos definidos en la fase de concepción: la especificación de los procesos del negocio, la lista de reserva del producto, las historias de usuarios, tareas de ingeniería. En la fase de desarrollo, que es donde se centra el objetivo de este trabajo. Se define cómo han de diseñarse las estructuras de datos, cómo ha de implementarse la función dentro de una arquitectura de *software*, cómo han de implementarse los detalles de los procedimientos, cómo han de caracterizarse las interfaces, cómo ha de traducirse el diseño en un lenguaje de programación y cómo ha de realizarse las pruebas. Cuenta con tres tareas: diseño del *software*, generación de código y prueba del *software* [17].

1.4.2. Lenguajes de programación

Un lenguaje de programación es el medio dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis. Se pone a disposición del programador para que este pueda comunicarse con los dispositivos *hardware* y *software* del ordenador que posea. Consta de un léxico, una sintaxis y una semántica, según el número de instrucciones que requiera para realizar una tarea específica, se clasifican en lenguaje de alto nivel y lenguaje de bajo nivel [18].

1.4.2.1. Java

Java posee una curva de aprendizaje muy rápida. Debido a su semejanza con C y C++, y dado que la mayoría de los desarrolladores los conocen, aunque sea de forma elemental, resulta muy fácil aprender Java, fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos Java fue diseñado para crear productos informáticos altamente fiables.

Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria. A nadie le gustaría ejecutar en su ordenador programas con acceso total a su sistema, procedentes de fuentes desconocidas. Así que se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.

Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados sistemas operativos dígase (Unix, Windows, Mac) opera también sobre diferentes estaciones de trabajo y distintas arquitecturas. El compilador de Java genera un formato intermedio indiferente a la arquitectura diseñada para transportar el código eficientemente a múltiples plataformas *hardware* y *software* [19]. Debido a estas características se escoge utilizar Java como lenguaje en el desarrollo de esta solución informática para que la misma sea multiplataforma y además porque la solución propuesta será integrada a la herramienta de diagramación del paquete Abad, que también usa este lenguaje en su desarrollo.

1.4.3. Marcos de trabajo para la gestión de interfaces de usuario en Java

Un marco de trabajo en el contexto de la programación es un conjunto de funciones o código genérico que realiza tareas comunes y frecuentes en todo tipo de aplicaciones (creación de objetos, conexión a base de datos, limpieza de cadenas, entre otras). Esto brinda una base sólida sobre la cual desarrollar aplicaciones concretas y permite obviar los componentes más triviales y genéricos del desarrollo. Los marcos de trabajo son construidos tomando como base los lenguajes orientados a objetos, esto permite una mejor modularización de los componentes y óptima reutilización de código. Además, en la mayoría de

los casos un marco de trabajo implementará uno o más patrones de diseño de *software* que aseguren la escalabilidad del producto [20].

1.4.3.1 AWT/Swing

Dos propuestas de bibliotecas de clases diseñadas para Java. La librería AWT (herramienta de ventana abstracta) se centra en las estructuras orientadas a eventos y viene incluida en la API (interfaz de programación) de Java como java.awt. Ya desde su primera versión, con lo que las interfaces generadas con esta biblioteca funcionan en todos los entornos Java disponibles. Presenta una jerarquía de clases que se divide en componentes y contenedores, con el fin de colocar los contenedores en la ventana y los componentes dentro de estos [21].

El paquete *Swing* hereda todo el manejo de eventos de AWT, pero además presenta la independencia de plataforma que este necesitaba. Los componentes de *Swing* utilizan la infraestructura de AWT, incluyendo su modelo de eventos, el cual rige cómo un componente reacciona a eventos tales como, eventos de teclado, ratón, y otros [21].

De esta forma, la unión de estos dos marcos de trabajo es muy provechosa para desarrolladores que buscan una mejor interfaz gráfica. Al estar AWT al mando de la gestión de eventos y el *Swing* encargado de los componentes agregados, que son las principales cualidades de estos marcos de trabajo, se logran excelentes resultados en el área del diseño. Debido a la necesidad de crear una interfaz agradable para el usuario, además de la necesaria interacción entre ellos, es que se utiliza este marco de trabajo en la implementación de la solución propuesta.

1.4.4. Ambiente de desarrollo integrado (IDE)

Se identificó para ser utilizado, cómo IDEs de desarrollo para la implementación de la herramienta el NetBeans 6.9, entorno muy utilizado en la programación mundial.

1.4.4.1 NetBeans

El desarrollo del sistema propuesto se realizará sobre el IDE NetBeans 6.9 basado en las funcionalidades ofrece, escribir, compilar, depurar y ejecutar programas. Trae soporte para crear aplicaciones en otros

lenguajes además de Java. Este es un producto libre y gratuito sin restricciones de uso. Es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans. Soporta el desarrollo de todos los tipos de aplicaciones Java. Entre sus principales componentes se encuentra un sistema de proyectos basado en control de versiones y refactorización. Facilita la disponibilidad de las mejoras de lenguaje y plataforma para los desarrolladores, acelerando el tiempo de desarrollo de aplicaciones. En muchas ocasiones los diseñadores buscan un ambiente gráfico agradable para sus aplicaciones y el NetBeans ofrece en su biblioteca AWT/*Swing* para gestionar interfaces de usuarios, un marco de trabajo que presenta muchas ventajas sobre sus similares de otros IDEs [22].

1.4.5. Lenguaje unificado de modelado (UML)

UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema informático. Representa decisiones y conocimientos sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está diseñado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar; pero está concebido para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos [23].

Las características más generales de UML son:

- Lenguaje de modelado orientado a objetos.
- Viabilidad en la corrección de errores.

La estandarización de un lenguaje de modelado es invaluable, ya que es la parte principal del proceso de comunicación que requieren todos los agentes involucrados en un proyecto informático. Si se quiere discutir un diseño con alguien más, ambos deben conocer el lenguaje de modelado y no así el proceso que se siguió para obtenerlo [23].

1.4.6 Herramienta de modelado

1.4.6.1. Visual Paradigm

Es una herramienta pensada para el ciclo completo del proyecto, dándole relevancia a todos los diagramas que genera. Permite dibujar diagrama de clases entre otros, permite además realizar código inverso, es decir, generar código desde diagramas y generar documentación. Tiene la capacidad de integrarse a entornos de desarrollo como el NetBeans IDE. Presenta cómo ventaja fundamental que a través de ella se pueden realizar prototipos de interfaz de usuarios que permiten tener una visión más cercana de cómo quedarían las interfaces del sistema [24]. Se utilizó el Visual Paradigm 8.0 para modelar el sistema propuesto, debido a las funcionalidades y ventajas que presenta el mismo, además de ser el propuesto por el centro de informatización universitaria (CENIA).

1.5 Conclusiones del capítulo

Con la realización de este capítulo se concluye que:

- El estudio del estado del arte permite conocer la forma en que se aplica la técnica de diagramación en el mundo.
- La definición de los conceptos asociados al problema de la investigación, a la disciplina de AI, la técnica de diagramación y el prototipado, permite un mejor entendimiento del negocio.
- El estudio de las herramientas homólogas permite demostrar la importancia, de desarrollar un módulo que contribuya a integrar los tres diagramas que crea el arquitecto de información, durante todo el proceso diagramación en una sola herramienta.
- La fundamentación de la metodología de desarrollo, los lenguajes y las herramientas propuestas permite un mejor desarrollo de la propuesta de solución.

Capítulo 2: Construcción del sistema

2.1 Introducción

A lo largo de este capítulo se realiza un estudio crítico y valorativo del análisis y diseño realizado y entregado por el analista. Se hace un análisis de la complejidad y funcionamiento de los algoritmos más importantes para la implementación del sistema. Se analizan los estándares de codificación utilizados y la arquitectura utilizada para este sistema. Además, se presenta la descripción de los principales paquetes del sistema, el modelo de despliegue definido para este trabajo de diploma.

2.2 Estándares de codificación

Los estándares de codificación son prácticas de programación que no están enfocadas a la lógica del programa, sino a su estructura, para facilitar la comprensión del sistema, el mantenimiento del código y asegurar que todos los programadores del proyecto trabajen de forma coordinada. El estándar debe abarcar todos los aspectos de la generación de código, además repercute directamente en lo bien que un programador comprende un sistema informático dado que aporta legibilidad al código fuente.

Usar estándares de codificación sólidos y realizar buenas prácticas de programación, con vistas a generar un código de alta calidad, es de gran importancia para la calidad del producto y para obtener un buen rendimiento del mismo [25]. Actualmente son utilizadas diversas notaciones para los diferentes lenguajes de programación, algunas de las más utilizadas son: notación Camel (camello), notación C, notación Húngara, notación para Constantes, entre otras. La notación es elegida a conveniencia del equipo de desarrollo, pero debe ser mantenida en un mismo programa.

Algunos lenguajes de programación sugieren una notación determinada, como Java y la notación Camel. Debido a que el lenguaje seleccionado es Java, será utilizada esta notación, en la implementación de los componentes del presente trabajo de diploma. La notación Camel consiste en escribir los identificadores con la primera letra de cada palabra en mayúsculas y el resto en minúscula, por ejemplo: EndOfFile (fin del archivo). Se denomina notación “Camel” porque los identificadores se asemejan a las jorobas de un camello.

Posee dos variantes

- UpperCamelCase, CamelCase o PascalCase: La primera letra es mayúscula.
- lowerCamelCase, camelCase o dromedaryCase: La primera letra es minúscula.

Se decidió utilizar la notación CamelCase en identificadores de clases, métodos y variables, siguiendo las recomendaciones de notación para este lenguaje de programación [26].

Para la sangría se determinó utilizar el estilo BSD (Distribución de Programas de Berkeley), debido a las ventajas de visibilidad que el mismo ofrece. Ejemplo de segmento de código aplicado:

```
public int BuscarElProyecto(String nombre)
  {
    int i = 0;
    while (i < listaProyectos.size() &&
        !listaProyectos.get(i).getNombreProyecto().equals(nombre))
    {
     i++;
}
  if (i != listaProyectos.size())
      return i;
    return -1;
  }
```

Figura 2.1 Segmento de código donde se aplica la notación Camel.

2.2.1. Tamaño de las líneas, llaves de apertura y cierre

En la implementación del sistema se seguirán las siguientes indicaciones:

- Se utilizará una identación sin tabulaciones, con un equivalente a cuatro espacios, para mantener integridad en las revisiones.
- El uso de las llaves { } se hará en una nueva línea.

- La longitud de las líneas de código es aproximadamente hasta ochenta caracteres para mantener la legibilidad del código.

Ejemplo:

```
1....TipoDato a = TipoDato b;
2
3....public void ejemplo ()
4.... {
5..../Bloque de instrucciones
6.....}
```

2.2.2 Estructuras de control

Permiten modificar el flujo de ejecución de las instrucciones de un programa, las estructuras de control pueden ser secuenciales, iterativas, cíclicas, entre otras. Los lenguajes de programación modernos tienen estructuras de control similares (*if, for, while, switch*) lo que varía de uno a otro es su sintaxis.

Para la uniformidad en el uso de las estructuras de control se establecen las siguientes normas:

- Entre la estructura de control y los paréntesis debe existir un espacio.
- Utilizar siempre llaves de apertura y cierre, incluso en situaciones en las que técnicamente son opcionales, esto aumenta la legibilidad y disminuye la probabilidad de errores lógicos.

Ejemplos:

```
1....if (condición)
2.... {
3 ..../Bloque de instrucciones
4....}
5....elseif (condición)
6.... {
7 ..../Bloque de instrucciones
8....}
```

2.3 XML

El XML (Lenguaje de Marcas Extensible) es una adaptación del SGML (Lenguaje de Marcas Estándar Generalizado) (ISO 8879). Un lenguaje de marcas es una forma de codificar un documento que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación. Los esquemas XML definen los elementos y atributos presentes en un documento y son la base de la organización de la información del mismo [27]. La solución debe recuperar información de ficheros XML, por lo que es necesario un esquema para garantizar la organización de la información en el archivo. A continuación se expone el esquema XML propuesto:

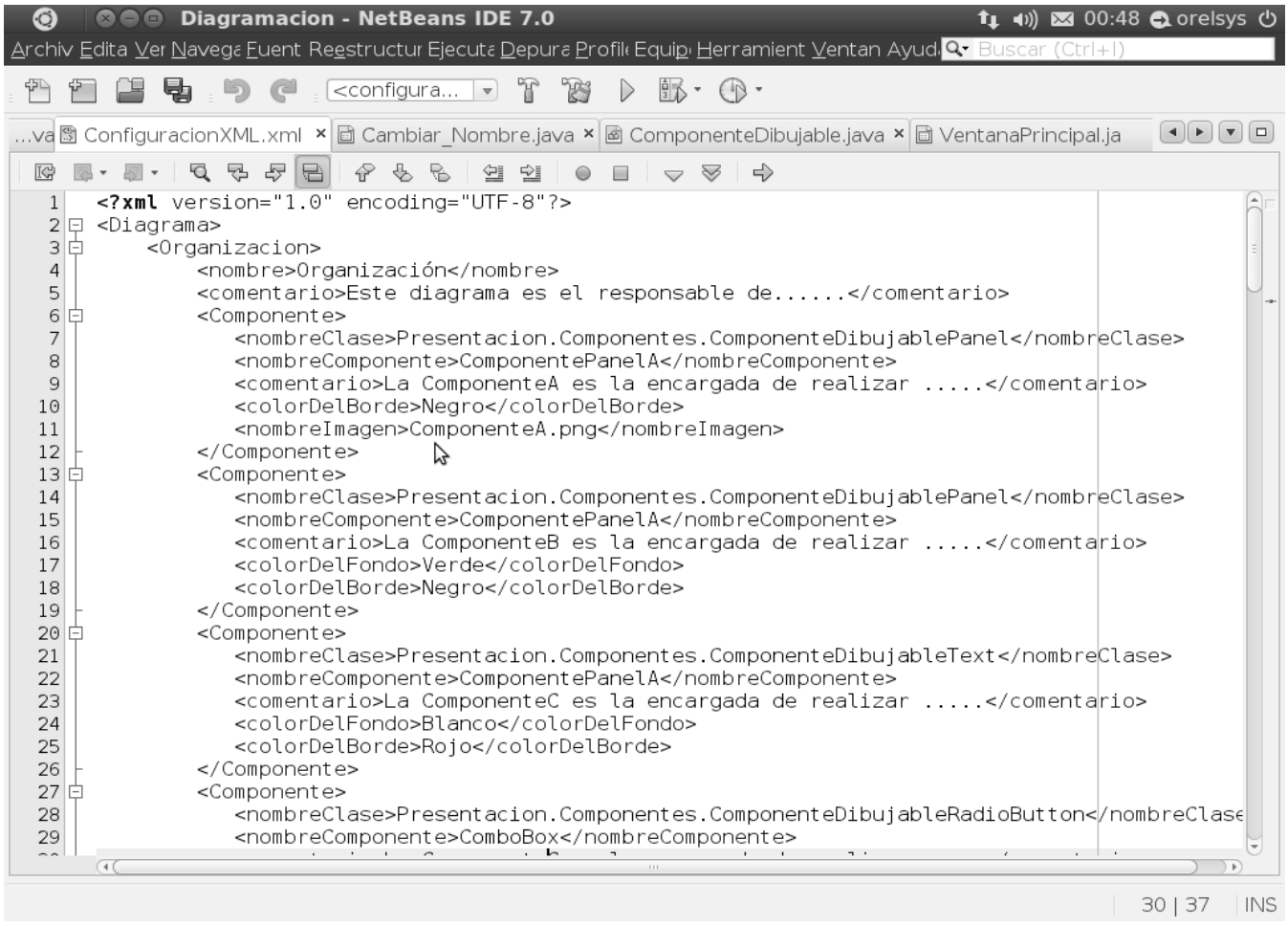


Figura 2.2 Esquema XML

2.4. Análisis crítico del diseño entregado

Después de obtener los artefactos de la planificación y el diseño, comienza la fase de implementación de la primera versión de la aplicación. Para ello es necesario un estudio preliminar de los resultados del diseño elaborado por el analista. Mediante la descripción detallada de las historias de usuarios, se puede establecer una estrategia de trabajo para guiar la implementación de los componentes. Luego se examina críticamente cada una de las funcionalidades necesarias y las clases persistentes relacionadas con estas. Garantizando la efectividad en el registro de información y comprobando que estén bien descritas, lo que facilita el trabajo a la hora de implementarlas.

Se verifica que lo expuesto en estas descripciones de las funcionalidades, sea posible de implementar. Se discuten las inconsistencias encontradas, con el objetivo de realizar los cambios necesarios para obtener un modelo de datos acorde a los objetivos del trabajo. Durante este proceso fueron encontradas incongruencias que se corrigieron para obtener un sistema mejor elaborado y de esta forma aumentar el nivel de aceptación de los usuarios. Finalmente, se determina que el diseño es el indicado y se puede continuar con las actividades de la fase de desarrollo, tomando en cuenta todas las modificaciones surgidas en el análisis expuesto en el presente acápite.

Luego de este análisis se concluye que:

- Las funcionalidades que permitirán mover hacia arriba y hacia abajo una página de un proyecto determinado fueron reemplazadas de la barra de herramienta e incorporada en la opción de clic derecho asignado a cada página en el árbol del proyecto.

- En la historia de usuario no queda bien especificada la funcionalidad correspondiente a los eventos de cada componente los cuales serán manejados desde la paleta de propiedades de cada componente, es decir, en la paleta de propiedades se hace clic sobre el ícono correspondiente a los eventos y acto seguido aparece la ventana donde se especificaran los eventos deseados.

- La funcionalidad correspondiente a guardar el proyecto fue modificada, donde se le incorporó la opción que permita guardar un diagrama por separado sin necesidad de tener que guardar el proyecto completo.

- Se definieron nuevas funcionalidades que no fueron concebidas por el analista en su estudio. Cómo la que permite exportar en formato HTML (lenguaje de marcado de hipertexto) un prototipo de interfaz de usuario realizado.

2.5 Patrón arquitectónico

Un patrón es una solución a un problema en un contexto determinado, codifica conocimiento específico acumulado por la experiencia en un dominio. Describe un problema que se instancia varias veces en un dominio determinado y propone su solución. En el caso de los patrones arquitectónicos proponen un esquema organizativo estructural para los sistemas informáticos [28].

2.5.2 Patrón arquitectónico de Tres Capas

Una arquitectura de *software* diseñada en capas consiste en la definición de niveles de abstracción, los cuales tienen una función específica permitiendo un diseño modular. Ello permite que sean creados sistemas con un bajo acoplamiento entre sus módulos o componentes. Una variante de este patrón muy utilizada es la de Tres Capas, mediante la cual se fracciona el sistema informático en la capa de presentación, la capa de lógica del negocio y la capa de acceso a datos. Una restricción de este patrón consiste, en que las capas inferiores no deben conocer ni hacer llamadas a procedimientos implementados en capas superiores, sino que las funcionalidades que ellas ofrecen, son accedidas desde niveles mayores [28].

La capa de presentación es la que se encarga de interactuar con el usuario mediante la interfaz de usuario. La capa lógica del negocio, llamada también cómo lógica de la aplicación, se encarga de realizar las tareas para las cuales está concebido el sistema. Es implementada utilizando un modelo orientado a objetos del dominio de la aplicación. Se encarga de controlar las operaciones de acuerdo con las reglas del negocio. La capa de acceso a datos es la que gestiona el almacenamiento de los datos, ya sea en una base de datos o en un fichero, así como la consulta a los mismos [29].

Ventajas de la arquitectura de tres capas:

- Fácil escalabilidad.
- Independencia de plataforma.
- Fácil comunicación.
- Reutilización del código.

Para el desarrollo de la solución que se presenta en este trabajo, se adoptará una arquitectura de tres capas. Debido a que la aproximación inicial del producto, se realizó utilizando este patrón, por lo que el desarrollo de este trabajo debe seguir el mismo principio. Fue seleccionado además debido a sus potencialidades para la reutilización y el aprovechamiento de los beneficios que brinda, además se tuvo en cuenta que muestra una clara separación entre las funciones desplegadas en capas. También permite la estandarización y proporciona una distribución clara del trabajo entre los miembros de un equipo de desarrollo.

2.6 Patrones de diseño

2.6.1 Patrones GRASP

Los patrones generales de *software* para la asignación de responsabilidades sus siglas en inglés GRASP (*General Responsibility Assignment Software Patterns*) describen los principios fundamentales de la asignación de responsabilidades a objetos [28]. En el desarrollo de esta solución se utilizan algunos patrones GRASP, los que se mencionan a continuación.

- **Experto** soluciona el problema de asignar una responsabilidad de forma general, al recomendar para esta función a la clase que maneje la información necesaria.
- **Creador** trata el problema de qué clase debe ser la encargada de instanciar a otra.
- **Bajo Acoplamiento** recomienda asignar las responsabilidades de tal forma que se le brinde soporte a una mayor reutilización y poca dependencia.
- **Alta Cohesión** aconseja mantener la clase lo más cohesionada posible para controlar la complejidad de la misma.
- **Controlador** ayuda a decidir quién se encarga de administrar un evento del sistema.

La siguiente tabla muestra, cómo se evidencia el uso de los diferentes tipos de patrones GRASP, mencionados anteriormente en el desarrollo de la solución del sistema.

Patrón	**Ejemplo de uso**
Experto	La clase ControladoraGraficaGeneral es la encargada de manejar todos los datos que tienen que ver con la creación de proyectos y que están dentro de la lógica del negocio.
Creador	La clase Proyecto contiene a la clase Pestannas, es una agregación, la almacena, tiene sus datos de inicialización y la usa.
Alta cohesión	La clase FabricaDeComponenteDibujable realiza una labor única dentro del sistema, y ninguna otra clase realiza sus funciones.
Bajo acoplamiento	Se utilizan las clases ControladaraGeneral, ControladoraGraficaGeneral para reducir la dependencia entre las clases.
Controlador	Se centralizan las actividades fundamentales en las Clases controladoras del sistema. Ejemplos: ControladoraGeneral, ControladoraGraficaGeneral.

Tabla1: Uso de los patrones GRASP en la solución

2.6.2 Patrones GoF

Los patrones de diseño, conocidos como patrones de cuatro bandas por sus siglas en inglés (GOF), se clasifican según el propósito para el cual han sido definidos en 3 grupos:

- **Creacionales:** solucionan problemas de creación de instancias. Ayudan a encapsular y abstraer dicha creación.
- **Estructurales:** solucionan problemas de composición (agregación) de clases y objetos.
- **De Comportamiento:** soluciones respecto a la interacción y responsabilidades entre clases y objetos, así como los algoritmos que encapsulan.

Para el desarrollo de los componentes de presentación del presente trabajo se utilizaron dos patrones GOF del tipo creacionales:

- Singleton: garantiza que una clase pueda ser instanciada una sola vez.
- Factory: tiene como objetivo devolver una instancia de múltiples tipos de objetos, normalmente, todos estos objetos provienen de una misma clase padre, mientras que se diferencian entre ellos por algún aspecto de comportamiento

La siguiente tabla muestra, cómo se evidencia el uso de algunos tipos de patrones GoF, mencionados anteriormente en el desarrollo de la solución del sistema.

Patrón	**Ejemplo de uso**
Singleton	Se utilizó para instanciar la clase ControladoraGraficaGeneral una sola vez, esto brinda la posibilidad de que todas las modificaciones que se realicen sean sobre esa instancia y no otra.
Factory	Se utilizó en la clase ComponenteDibujable que en este caso es la clase padre, pero al instanciarla en realidad el objeto lo que va a contener, es una instancia de una de sus clases hijas, las que pueden ser ComponenteDibujableButton, ComponenteDibujableJComboBox, ComponenteDibujableMenu, ComponenteDibujableRadioButton, entre otros.

Tabla 2: Uso de patrones GoF en la solución.

2.7 Modelo del diseño

De acuerdo con lo que plantea la arquitectura en tres capas, la cual organiza las clases en tres paquetes, se organizaron las clases en la solución que plantea este trabajo de diploma. Cada uno de

los paquetes se ha dividido a la vez en sub-paquetes de acuerdo con las funcionalidades de las clases contenidas. Cada paquete representa una capa de la arquitectura y está construido sobre su predecesor. Las clases pertenecientes a una capa están relacionadas con las clases de la capa inmediata inferior y desconocen a las clases de las capas superiores, ver Figura 2.3.

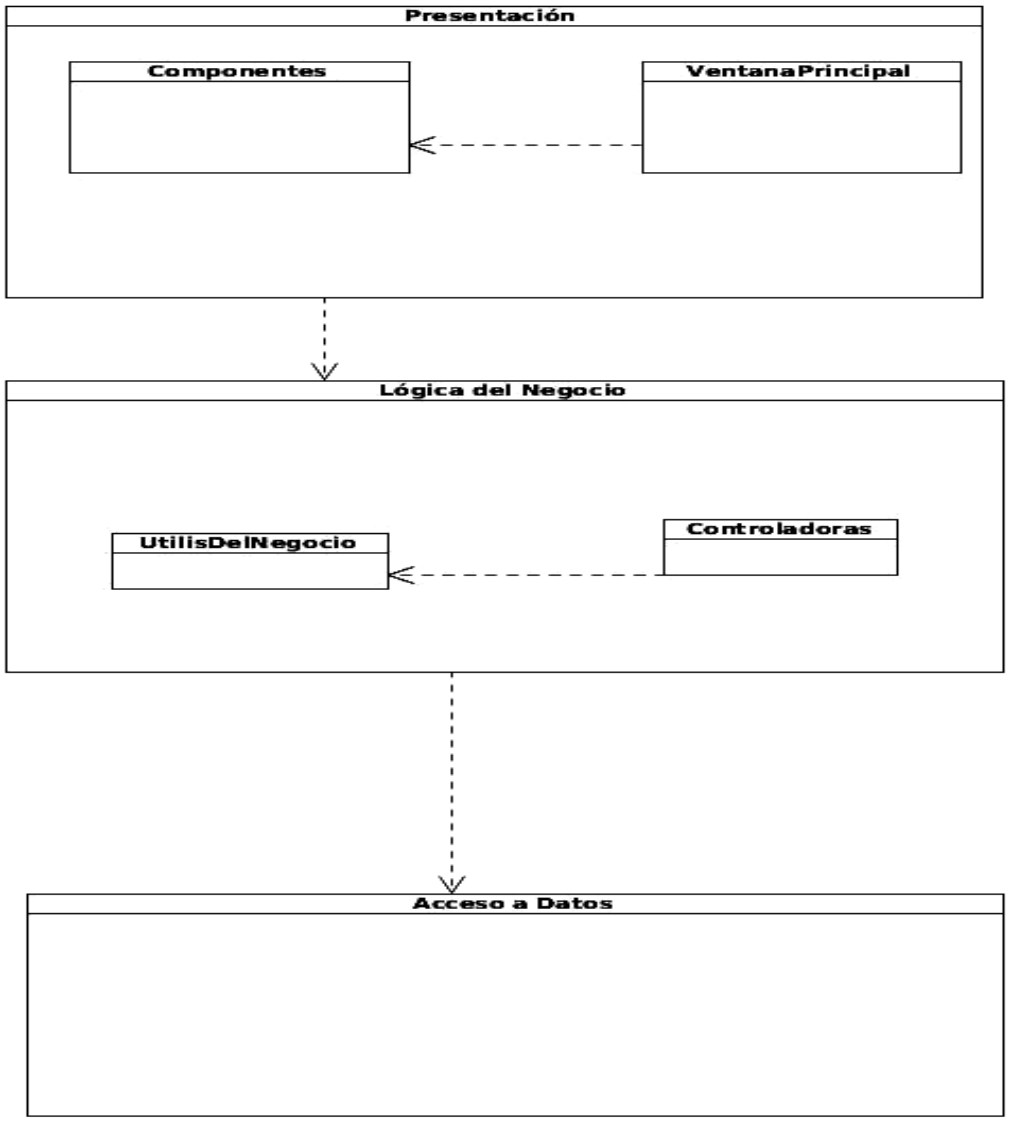


Figura # 2. 3 Diagrama de Paquetes

2.7.1 Acceso a datos

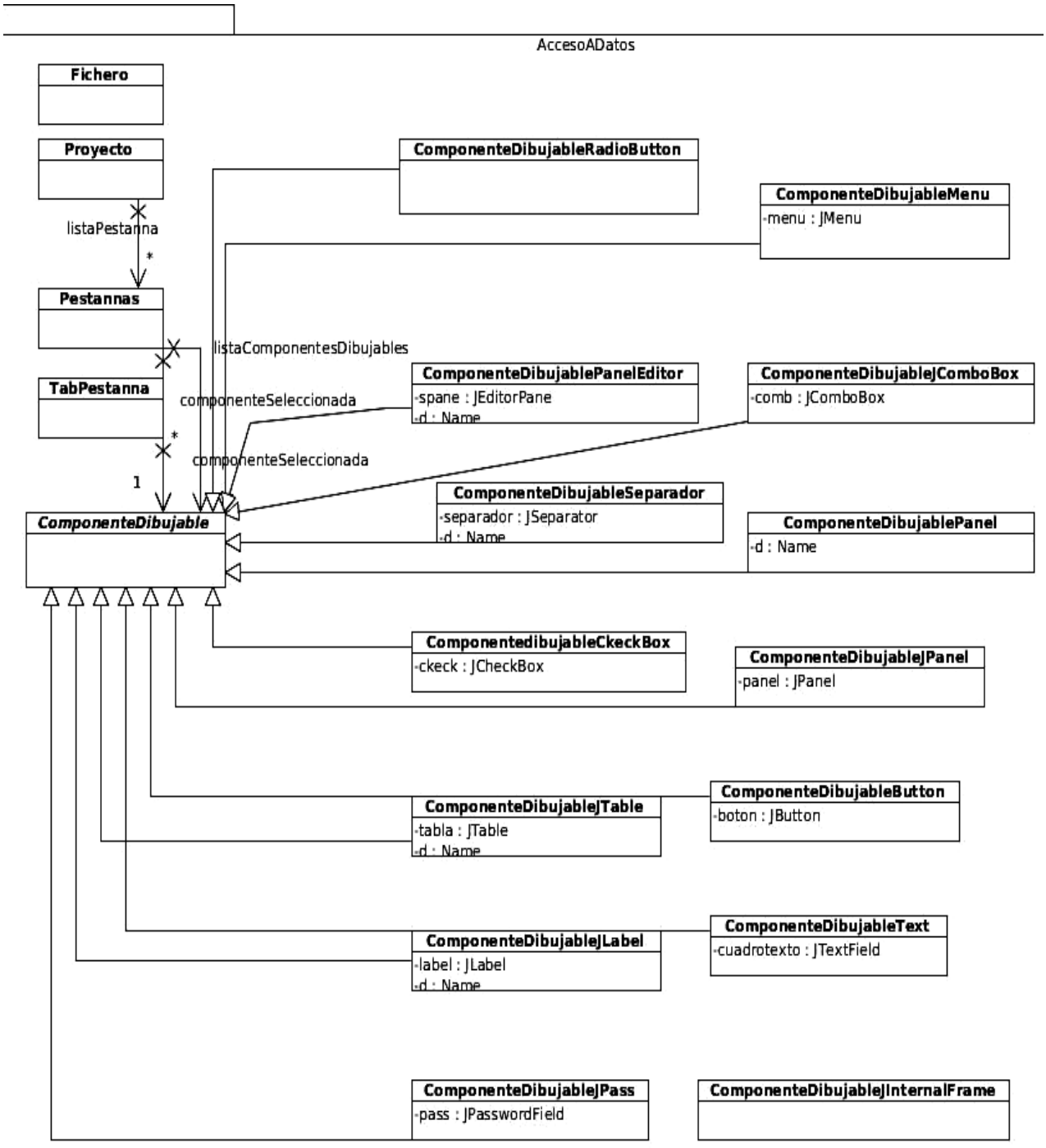
La capa de acceso a datos es la encargada de la persistencia y recuperación de objetos, específicamente de la interacción de la aplicación con los ficheros de almacenamiento. Los que pueden contener ficheros en forma de objeto serializado o XML. Esta capa presenta una clase fichero que es la que posee la información persistente de un proyecto. El paquete de acceso a datos mantiene un bajo acoplamiento con las entidades del negocio. En esta capa aparece una serie de clases entidades, las que contienen los datos persistentes de cada elemento que se maneja en la aplicación como son: Proyecto, Pestanna, TabPestanna, Componentes de presentación, entre otros que quedan evidenciados en el diagrama correspondiente a este paquete en la Figura 2.4.

2.7.2 Capa lógica del negocio

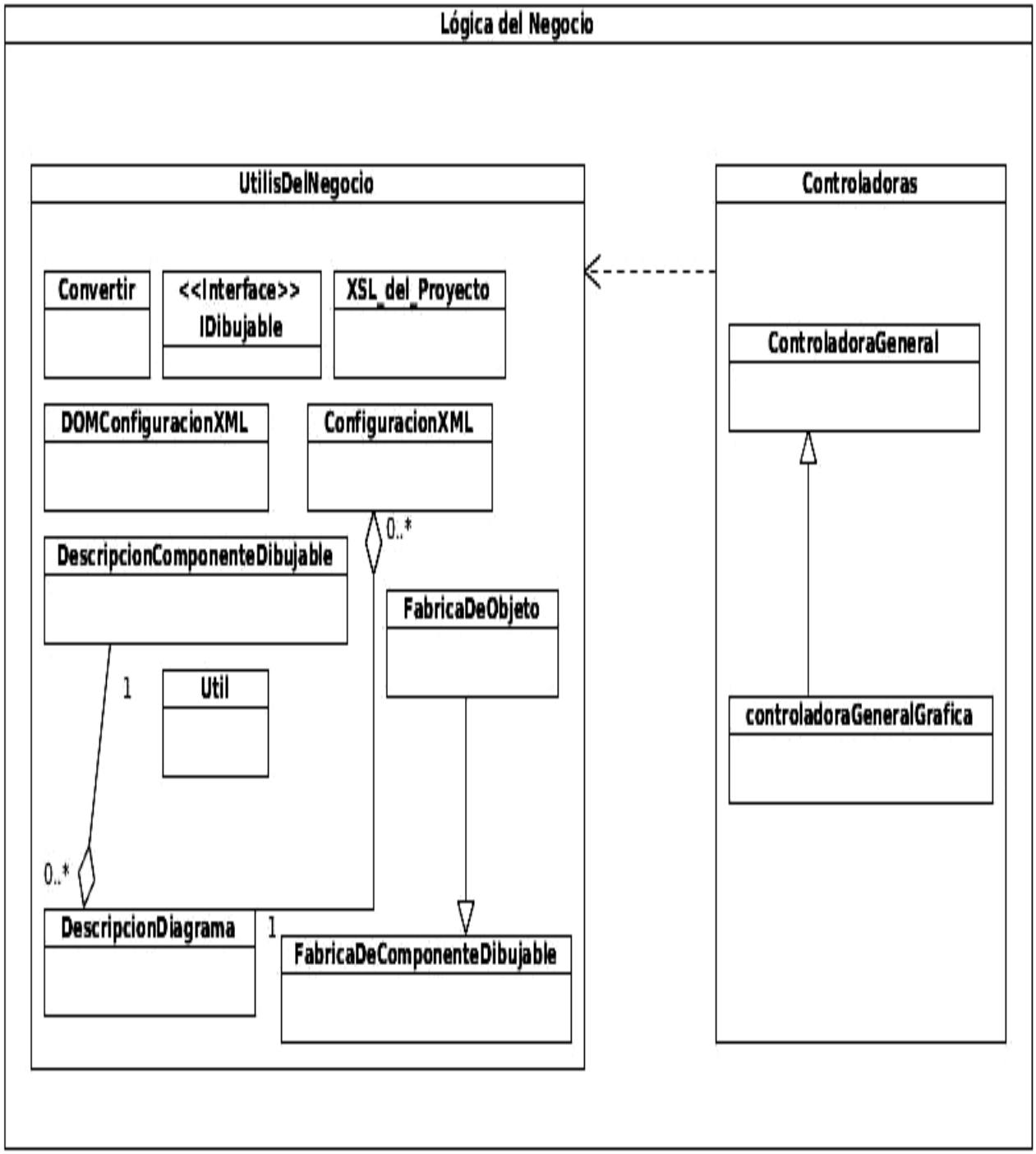
En la capa lógica del negocio es donde se establece todas las reglas que se deben cumplir, es donde se gestionan todas las funcionalidades para importantes del sistema. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y con la capa de datos, para solicitar almacenar o cargar los datos en ficheros o en archivo XML. En el diseño realizado para esta solución, la capa lógica del negocio se divide en otras 2 subcapas menos densas, agrupando las clases en dependencia de la función que tienen en el sistema. Cómo se muestra a continuación.

- Paquete (Controladoras): es la base de la capa lógica del negocio. Las clases de este paquete son las encargadas de gestionar las funcionalidades para las que fue concebido el sistema. Las clases que posee este paquete son, ControladoraGeneral y ControladoraGeneralGrafica.
- Paquete (UtilesDelNegocio): Posee una serie de clases útiles, las que son utilizadas para realizar alguna función específica dentro del sistema, ya sea definir la configuración del fichero XML, definir algunos enumerativos utilizados en el sistema entre otras.

La figura 2.5 muestra cómo quedaría esta capa.



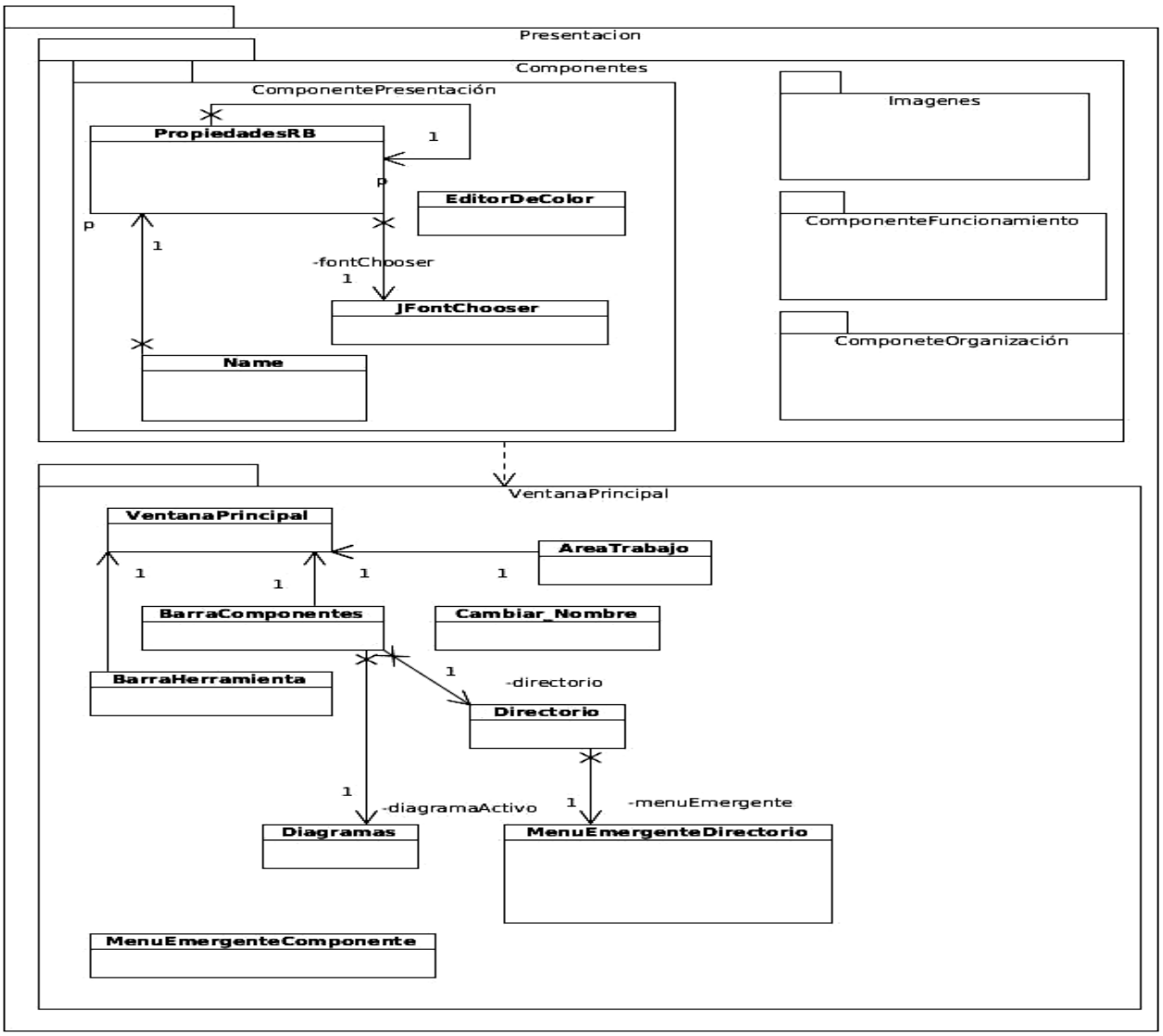
Figura# 2.4 Diagrama de clases del diseño de la capa de acceso a datos



Figura# 2.5 Diagrama de clases del diseño de la capa de lógica del negocio

2.7.3 Capa presentación

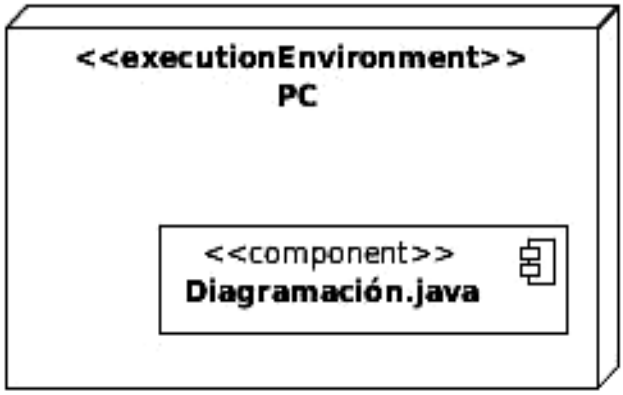
La capa de presentación es donde se encuentra todo lo relacionado con la interfaz visual del sistema, la cual es la encargada de interactuar directamente con el usuario. Esta capa está dividida en dos sub paquetes, el paquete VentanaPrincipal, que contiene relacionado con la interfaz principal del sistema. El paquete Componentes que contiene los formularios y ventanas correspondientes a los componentes de presentación. La figura 2.6 muestra el diseño de este paquete.



Figura# 2.6 Diagrama de clases del diseño de la capa de presentación

2.8 Modelo de despliegue

El modelo de despliegue es un modelo de objetos que describe la organización física del sistema partiendo de cómo se distribuye la funcionalidad entre los nodos de cómputo [30]. La presente solución es una aplicación para entornos de escritorio, que no interactúa en ningún momento con una base de datos. Por lo que será ejecutada en una estación de trabajo, con una computadora que posea 256 megabytes de memoria o más, y 120 gigabytes de disco duro. La Figura 2.7 muestra el modelo de despliegue de la solución.



Figura# 2.7 Modelo de despliegue

2.10 Conclusiones del capítulo

Con la realización de este capítulo se concluye que:

- La valoración crítica realizada al análisis y diseño, permite obtener el resultado deseado, contribuyendo a mejorar el proceso de implementación.
- La definición de los patrones de diseños y de arquitectura permite la obtención de un diseño exitoso.
- El esbozo de los módulos utilizados en la solución con las funcionalidades principales facilitó el entendimiento de la arquitectura del sistema.

Capítulo 3: Pruebas del sistema

3.1 Introducción al capítulo

En el presente capítulo se va a documentar la realización de las pruebas que se aplicarán al sistema. Se realizarán pruebas de unidad y las pruebas de aceptación para validar el sistema informático. Se construirán los diseños de caso de pruebas basados en las historias de usuario elaboradas en la etapa de planificación y quedarán documentadas las no conformidades detectadas en cada una de las iteraciones.

3.2 Pruebas del software

Las pruebas de *software* son un conjunto de herramientas y técnicas que evalúan el desempeño de un programa. En todo proceso de desarrollo de aplicaciones, es indispensable la presencia de un proceso de pruebas de *software*, que garantice el buen funcionamiento y la calidad del producto final, según Pressman: “Las pruebas del *software* son un elemento crítico para la garantía de calidad del producto y representa una revisión final de las especificaciones, del diseño y de la codificación” [31].

Las pruebas de *software* permiten conocer hasta qué punto las funciones del sistema informático, operan de acuerdo con las especificaciones y requisitos del cliente. Estas involucran las operaciones del sistema, evaluando los resultados bajo condiciones específicas, lo que hace que la realización de pruebas a los programas constituya un factor de vital importancia. El objetivo de las pruebas, expresado de forma sencilla, es encontrar el mayor número posible de errores en un período de tiempo moderado, y con la mínima cantidad de esfuerzo. Un proceso de prueba completo debe garantizar además, que los defectos encontrados sean corregidos antes de entregar el producto al cliente [31].

En concordancia con lo que establece la metodología utilizada para el desarrollo de la aplicación, las pruebas que serán aplicadas al sistema serán:

- Pruebas de unidad
- Pruebas de aceptación

3.3 Pruebas de unidad

Es la escala más pequeña de las pruebas, está basada en la funcionalidad de los módulos del programa, cómo funciones, procedimientos y módulos de clases [31]. Es la prueba enfocada a los elementos probables más pequeños del *software*. Es aplicable a componentes representados en el modelo de implementación, para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera.

Los casos de pruebas que se diseñen para este nivel de pruebas, deben descubrir errores como:

- Comparaciones entre tipos de datos distintos.
- Operadores lógicos o procedencia incorrecta.
- Igualdad esperada cuando los errores de precisión la hacen poco probable.
- Variables o comparaciones incorrectas.
- Fallo de salida cuando se encuentra una iteración divergente.

Para la realización de este tipo de prueba se analizaron 10 fragmentos de código, los cuales fueron escogidos de diferentes clases en cada una de las tres capas, con el objetivo de abarcar la mayor cantidad de clases posibles. Una vez realizada esta prueba se concluye que todas funcionan correctamente, pero esto no garantiza el correcto funcionamiento del sistema, por lo que se propone la realización de pruebas que abarquen más código. En la Figura 3.1 se muestra uno de los fragmentos de código probados mediante el método de camino básico.

El completo entendimiento de esta prueba se apoya en un grafo, que se forma con los números (para crear el grafo se transforman en nodos) que aparecen a la izquierda en la figura anterior. En grafo están presentes todos los caminos que pueden guiar la trayectoria del módulo o unidad, de manera que todos sus nodos, excepto el primero y el último, deben tener al menos una entrada y una salida. En la figura 3.2 se presenta el grafo perteneciente a esta prueba.

```
private void EliminarPestanna()
        {
   1-   if (listaProyectos.get(indiceProyActivo).getListaPestanna().size() == 1)
         {
   2-   JOptionPane.showMessageDialog(null, "La página seleccionada no puede ser eliminada, pues cada proyecto
debe tener mínimo una página.");
       }
       else
       {
   3-   int opcion = JOptionPane.showOptionDialog(null, "¿Está seguro que desea eliminar la página?", "Eliminar Página",
JOptionPane.YES_NO_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE, null, null, null);
   4-   if (opcion == 0)
        {
   5-   String nombrePag = barraComponente.getDirectorio().getSelectionPath().getLastPathComponent().toString();
             barraComponente.getDirectorio().EliminarNodo(null);
             Pestannas p = new Pestannas(nombrePag);
             p.setNombreProyecto(listaProyectos.get(indiceProyActivo).getNombreProyecto());
             area.getTabPestannas().EliminarPagina(p);

             listaProyectos.get(indiceProyActivo).EliminarLaPestanna(nombrePag);
   6-       }
   7-    }
   8- }
```

Figura 3.1 Fragmentos de códigos probados mediante el método de camino básico

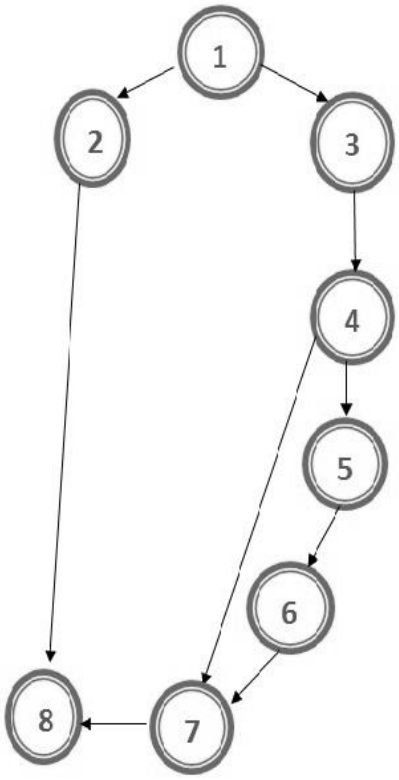


Figura 3.2 Grafo realizado para aplicar el método de camino básico

Este grafo se usa para calcular la complejidad ciclo matica del módulo a probar. Para ello son usadas tres vías distintas, cada una de ellas se define por una fórmula, que independientemente de cuál de ellas se use, el resultado siempre debe ser el mismo.

Primera fórmula:

V (G) = (A - N) + 2 (donde A es el número aristas del grafo y N el número de nodos.)

V (G) = (9-8) + 2

V (G) = 3

Segunda fórmula:

V (G) = P + 1 (donde P es la cantidad de nodos desde los cuales se pueden tomar dos caminos.)

V (G) = 2 + 1

V (G) = 3

Tercera fórmula:

V (G) = R (donde R es la cantidad de regiones)

V (G) = 3

Luego de la aplicación de las tres fórmulas se puede apreciar que el resultado obtenido en todos los casos es el mismo, 3. Esto sirve para determinar que la cantidad de caminos básicos que pueden ser tomados, es 3 en total. Es decir, el procedimiento puede tomar por 3 caminos distintos. De esta forma, se identifica que para este fragmento de código, se debe realizar un caso de prueba, con tres escenarios, donde cada uno de ellos corresponde a un camino identificado.

- Caminos que pueden ser tomados: 1, 2, 8; 1, 3, 4, 7, 8; 1, 3, 4, 5 , 6, 7, 8.

El siguiente paso es preparar los casos de prueba, para la ejecución de cada camino del conjunto básico, asegurando que los datos de entrada necesarios para ejecutar la funcionalidad son correctos. Para luego obtener, al terminar cada prueba, una respuesta como resultado a la correcta evaluación de la función.

- **Camino:** 1,7

  Caso de prueba: Eliminar una Pestaña.

  Entrada: Lista de pestañas vacía.

  Resultado: Se muestra un mensaje, aclarando que debe existir al menos una pestaña.

- **Camino:** 1,3,4,7,8

  Caso de prueba: Eliminar una pestaña.

  Entrada: Lista de pestañas, con la pestaña que se desea eliminar.

  Resultado: Se visualiza una ventana de confirmación, donde se selecciona la opción “Cancelar”.

- **Camino:** 1, 3, 4, 5, 6, 7, 8.

  Caso de prueba: Eliminar una pestaña.

  Entrada: Lista de pestañas, con la pestaña que se desea eliminar.

  Resultado: Se visualiza una ventana de confirmación, donde se selecciona la opción “Cancelar”.

3.4 Pruebas de aceptación

Las pruebas de aceptación constituyen la validación de la aplicación por parte del usuario final. Según Pressman: “la validación del *software* se consigue mediante una serie de pruebas de caja negra, que demuestran la conformidad con los requisitos.” [32]. Se realizarán pruebas funcionales para verificar los dominios de entrada y salida del programa, y así descubrir errores en la funcionalidad y comportamiento del mismo. Para realizar este tipo de pruebas a la aplicación, se utilizará el método de pruebas de caja negra que se centran en los requisitos funcionales del *software*. Estas pruebas se llevan a cabo sobre la interfaz del sistema informático, mediante un conjunto de datos de entrada, que ejercitan completamente todos los requisitos funcionales de un programa. Los clientes son los responsables de verificar que los resultados de estas pruebas sean correctos. Para la realización de las mismas se escogieron usuarios del proyecto Abad, con un conocimiento básico en la técnica de diagramación, dentro del proceso AI. Se utiliza para la ejecución de estas pruebas los diseños de casos de prueba (DCP).

3.4.1 Diseños de casos de pruebas (DCP)

Los casos de prueba pretenden demostrar, que las funciones del *software* son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto. Los DCP son creados sobre la base de las historias de usuario y usados por los clientes para comprobar que las mismas cumplen con su cometido. En este acápite se incluirán los DCP, realizados a partir de las historias de usuarios críticas del sistema, los demás quedarán adjuntados en el Anexo 1.

DCP 1.Gestionar proyecto

SC 1.1 Crear proyecto

Escenario	Descripción	Respuesta del sistema	Flujo central
Crear proyecto correctamente	El sistema permite crear un nuevo proyecto	Se muestra la ventana que posee la vista del proyecto.	1.Clic en el menú Archivo 2. Seleccionar la opción "Nuevo proyecto".
			1-Clic derecho sobre nodo padre, del árbol del proyecto. 2-Clic en la opción Adicionar proyecto.

SC 1.2 Eliminar proyecto

Escenario	Descripción	Respuesta del sistema	Flujo central
Eliminar proyecto correctamente.	El sistema permite eliminar correctamente un proyecto.	La herramienta elimina el proyecto y se muestra un mensaje de confirmación "¿Está seguro de que desea eliminar el proyecto? ".	1. Clic derecho encima del proyecto que se desea eliminar. 2. Seleccionar la opción "Eliminar proyecto". 3. Clic en el botón "Aceptar".
Cancelar.	La herramienta no elimina el proyecto.	La herramienta no elimina el proyecto y se muestra un mensaje de confirmación "¿Está seguro de que desea eliminar el proyecto?".	1. Clic derecho encima del proyecto que se desea eliminar.

			2. Seleccionar la opción “Eliminar proyecto”. 3. Clic en el botón “Cancelar”.

SC 1.3 Cambiar nombre

Escenario	Descripción	Nombre	Respuesta del sistema	Flujo central
Cambiar nombre correctamente.	El sistema permite cambiarle el nombre al proyecto.	V (pepe)	La herramienta permite salvar el proyecto correctamente en la dirección deseada.	1. Clic derecho encima del proyecto. 2. Seleccionar en las opciones que se muestra “Cambiar nombre”. 3. Se muestra una ventana. 4. Se inserta el nombre. 5. Oprimir el botón “Aceptar”.
Nombre incorrecto	El usuario intenta ponerle un nombre que ya existe al proyecto.	I (Pepito)	La herramienta muestra en mensaje “El proyecto con ese nombre ya existe”.	1. Clic derecho encima del proyecto. 2. Seleccionar en las opciones que se muestra “Cambiar nombre”. 3. Se muestra una ventana. 4. Se inserta el nombre.

				5. Oprimir el botón “Aceptar”.
Nombre vacío	El usuario deja en blanco el nombre del proyecto.	I (Vacío)	La herramienta no cambia el nombre al proyecto y muestra un mensaje “El campo nombre no puede estar vacío”	1. Clic derecho encima del proyecto. 2. Seleccionar en las opciones que se muestra “Cambiar nombre”. 3. Se muestra una ventana. 4. Se inserta el nombre. 5. Oprimir el botón “Aceptar”.

SC 1.4 Exportar el proyecto como HTML

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
Exportar el proyecto como HTML	La herramienta permite exportar el proyecto como HTML.	La herramienta permite seleccionar la dirección para exporta el proyecto a HTML.	1. Clic derecho encima del proyecto. 2. Seleccionar en las opciones que se muestra “Exportar el proyecto como HTML”. 3. Se muestra una ventana para escoger la dirección deseada. 4. En la misma ventana

			brinda la opción para ponerle el nombre. 5. Oprimir el botón "Aceptar".
Exportar el proyecto como HTML incorrectamente.	La herramienta no permite exportar a HTML a una dirección que se especifique.	La herramienta no permite exportar el proyecto a HTML muestra un mensaje "Error al especificar la dirección".	1. Clic derecho encima del proyecto. 2. Seleccionar en las opciones que se muestra "Exportar el proyecto como HTML". 3. Muestra una ventana para poner la dirección. 4. Se escoge la dirección. 5. Se oprime el botón "Aceptar".

SC 1.5 Salvar proyecto

Escenario	Descripción	Respuesta del sistema	Flujo central
Salvar proyecto correctamente.	La herramienta permite salvar el proyecto correctamente.	La herramienta permite salvar el proyecto correctamente en la dirección deseada.	1. Clic derecho encima del proyecto. 2. Seleccionar en las opciones que se muestra "Salvar proyecto".

			3. Selecciona la dirección. 4. Ponerle el nombre. 5. Oprimir el botón “Guardar”.
Cancelar	Es cancelada la operación de guardar el proyecto.	No se salva el proyecto	1. Clic derecho encima del proyecto. 2. Seleccionar en las opciones que se muestra “Salvar proyecto”. 3. Selecciona la dirección. 4. Ponerle el nombre. 5. Oprimir el botón “Cancelar”.
Salvar proyecto incorrectamente con un nombre ya existente.	La herramienta no permite exportar el proyecto como HTML.	La herramienta no salva el proyecto y muestra un mensaje “Error al especificar la dirección”.	1. Clic derecho encima del proyecto. 2. Seleccionar en las opciones que se muestra “Salvar proyecto”. 3. Selecciona la

			dirección. 4. Ponerle un nombre existente. 5. Oprimir el botón “Guardar”.

Descripción de las variables

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Entero y String	No	
2	Nombre de la carpeta	String	No	No admite caracteres extraños

DCP 2. Gestionar página

SC 2.1 Crear página

Escenario	Descripción	Respuesta del sistema	Flujo central
Crear una página correctamente	El sistema permite crear una nueva página	Se muestra la venta que muestra la vista da la página.	1.Clic en el menú Archivo 2. Seleccionar la opción “Nueva página”.
			1-Clic en el ícono que permite crear una página en la barra de herramienta.

			1-Clic derecho en el árbol del proyecto. 2-clic en la opción “Adicionar página”.
Crear página incorrectamente	El sistema no crea una nueva página	Se muestra un mensaje “Debe existir al menos un proyecto creado”.	1.Clic en el menú Archivo 2. Seleccionar la opción” Nueva página”.
			1-Clic en el ícono que permite crear una página en la barra de herramienta
			1-Clic derecho en el árbol del proyecto. 2-clic en la opción “Adicionar página”.

SC 2.2 Eliminar página

Escenario	Descripción	Respuesta del sistema	Flujo central
Eliminar página correctamente.	El sistema permite eliminar correctamente una página.	La herramienta muestra un mensaje de confirmación “¿Está seguro de que desea eliminar la página?”.	1. Clic derecho encima de la página que se desea eliminar. 2. Seleccionar la opción “Eliminar página”.

			3. Clic en el botón “Aceptar”.
Cancelar.	El usuario cancela la operación de eliminar el proyecto.	La herramienta no elimina la página y se muestra un mensaje de confirmación “¿Está seguro de que desea eliminar el proyecto?”.	1. Clic derecho encima de la página que se desea eliminar. 2. Seleccionar la opción “Eliminar página”. 3. Clic en el botón “Cancelar”.

SC 2.3 Cambiar nombre

Escenario	**Descripción**	**Nombre**	**Respuesta del sistema**	**Flujo central**
Cambiar nombre correctamente.	El sistema permite cambiarle el nombre a la página.	V (pepe)	La herramienta le cambia el nombre a la página correctamente.	1. Clic derecho encima de la página. 2. Seleccionar en las opciones que se muestra “Cambiar nombre”. 3. Se muestra una ventana. 4. Se inserta el nombre. 5. Oprimir el botón “Aceptar”.

Nombre incorrecto	El usuario intenta ponerle un nombre que ya existe a la página.	I (Pepito)	La herramienta muestra en mensaje “La página con ese nombre ya existe”.	1. Clic derecho encima de la página. 2. Seleccionar en las opciones que se muestra “Cambiar nombre”. 3. Se muestra una ventana. 4. Se inserta un nombre. 5. Oprimir el botón “Aceptar”.
Nombre vacío	El usuario deja el nombre en blanco.	I (Vacío)	La herramienta no cambia el nombre a la página y muestra un mensaje “El campo de nombre no puede estar vacío”.	1. Clic derecho encima de la página. 2. Seleccionar en las opciones que se muestra “Cambiar nombre”. 3. Se muestra una ventana. 4. Deja el campo en blanco. 5. Oprimir el botón “Aceptar”.

SC 2.4 Exportar página como HTML

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
Exportar página como	La herramienta permite	La herramienta permite	1. Clic derecho encima

HTML	exportar la página como HTML.	seleccionar la dirección para exportar la página HTML.	de la página. 2. Seleccionar en las opciones que se muestra “Exportar la página a HTML”. 3. Se muestra una ventana para escoger la dirección deseada. 4 .Oprimir el botón “Aceptar”.
Exportar la página como HTML incorrectamente.	La herramienta no permite exportar a HTML a una dirección que se especifique.	La herramienta no permite exportar la página a HTML muestra un mensaje “Error al especificar la dirección”.	1. Clic derecho encima de la página. 2. Seleccionar en las opciones que se muestra “Exportar la página a HTML”. 3. Muestra una ventana para poner la dirección. 4. Se oprime el botón “Aceptar”.

SC 2.5 Salvar página

Escenario	Descripción	Respuesta del sistema	Flujo central
Salvar página correctamente.	La herramienta permite salvar la página correctamente.	La herramienta permite salvar la página correctamente en la dirección deseada.	1. Clic derecho encima de la página. 2. Seleccionar en las opciones que se muestra “Salvar página”. 3. Selecciona la dirección. 4. Ponerle el nombre. 5. Oprimir el botón “Guardar”.
Cancelar	El usuario cancela la operación de salvar la página.	No se salva la página	1. Clic derecho encima de la página. 2. Seleccionar en las opciones que se muestra “Salvar página”. 3. Selecciona la dirección. 4. Ponerle el nombre. 5. Oprimir el botón “Cancelar”.
Salvar página	La herramienta no	La herramienta no salva	11. Clic derecho encima

incorrectamente con un nombre ya existente.	permite salvar la página	la página y muestra un mensaje ¨Error al especificar la dirección ¨	de la página. 2. Seleccionar en las opciones que se muestra salvar página 3. Selecciona la dirección. 4. Ponerle el nombre. 5. Oprimir el botón Guardar.

Descripción de las variables

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Entero y String	No	
2	Nombre de la carpeta	String	No	No permite caracteres extraños.

3.4.2 Resultados de las pruebas

Al sistema se le realizaron 3 iteraciones de pruebas, con ayuda de los diseños de casos de pruebas, para verificar el correcto funcionamiento de la aplicación. En la primera iteración de pruebas de 25 requisitos funcionales implementados, se identificaron 13 no conformidades, de las cuales se le dio solución a 11. En la segunda iteración de 40 requisitos implementados, se identificaron 18 no conformidades, las que fueron corregidas en su totalidad. En la tercera iteración de 11 requisitos funcionales implementados, se detectaron 3 no conformidades, las que se corrigieron en su totalidad.

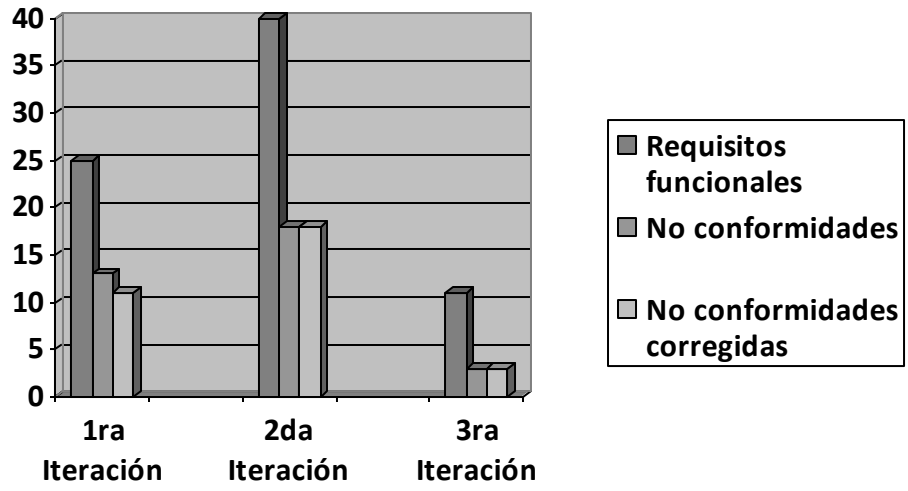


Figura 3.3 Resultados de las pruebas

3.5 Conclusiones de capítulo

Con la realización de este capítulo se concluye que:

- La aplicación de las pruebas realizadas al sistema, permitió detectar y corregir no conformidades, que pudieran atentar en contra del correcto funcionamiento del sistema final.
- El diseño de casos de pruebas permitirá corregir errores que puedan existir en el sistema.

Conclusiones

La presente investigación ha mostrado los sistemas existentes, que se utilizan para realizar la técnica de diagramación, y qué tipos de diagramas realizan. Mostrando la necesidad de una aplicación que integre los tres tipos de diagramas que realiza el arquitecto de información, durante todo el proceso de diagramación, para facilitar y agilizar el trabajo de este. En este marco y al finalizar el desarrollo de este trabajo de diploma, se concluye que:

- El estudio de soluciones homólogas permitió definir las características del sistema desarrollado, y demostró la necesidad de implementar los componentes de presentación para la herramienta de diagramación. Para de esta forma contribuir al desarrollo de una herramienta, que integre los componentes necesarios para la construcción de los diagramas, que realiza el arquitecto de información, en la técnica de diagramación.

- La aplicación informática que se realizó, contribuye al desarrollo de una herramienta que permitirá la integración de los tres tipos de diagramas pertenecientes a la técnica de diagramación (organización, flujo, presentación).

- La valoración crítica del análisis y diseño realizado por el analista, permitió refinar las funcionalidades, para mejorar las características del sistema acorde a las expectativas del usuario.

- El sistema implementado ayudará a agilizar el flujo de los procesos de arquitectura de información en la UCI, y así lograr una mayor calidad en la organización y estructuración de los contenidos en las soluciones informáticas.

Recomendaciones

El constante avance que experimenta el mundo del *software* es la premisa fundamental para continuar la perfección del sistema elaborado. Con este objetivo se recomienda para próximas versiones:

- Implementar nuevos componentes que impliquen un acabado mayor, en la realización de los diagramas de presentación.
- Mejorar la interfaz visual del producto.
- Implementar funcionalidades que permita exportar un prototipo como una imagen y como PDF.
- Trabajar en el desarrollo de los eventos, perteneciente a cada componente.
- Realizar un manual de usuario que brinde mayores facilidades de uso del sistema.

Bibliografía referenciada

1. **Gutiérrez, Mario Pérez-Montoro.** *Arquitectura de la información en entornos web.* Gijón : Trea, 2010. (En prensa).

2. **Sanchez de Bustamante, Antonio Montes de Oca.** *A.Arquitectura de información y usabilidad: nociones básicas para los profesionales de la información. 2004.* Ciudad Habana : Acimed, 2004. Disponible en: http://bvs.sld.cu/revistas/aci/vol12_6_04/aci04604.htm.

3. **Garret, J.J.** A visual vocabulary for describing information architecture and interaction desing. [En línea] En línea. [Citado el: 16 de febrero de 2011.] http://www.jjg.net/ia/visvocab.

4. **León, Rodrigo Ronda.** [En línea] 2007. http://www.nosolousabilidad.com/articulos/tecnicas_ai.htm.

5. **Perez, Meritxell Ramon.** *Evaluación de herramientas para prototipado de sistemas interactivos.* Lleida : s.n., 2010.

6. **Granollers, Tony.** *Una metodología que integra la Ingeniería del Software, la interacción persona-ordenador y la accesibilidad en el contexto de los equipos de desarrollo multidisciplinares.* Lleida : Univertitat de Lleida, 2004.

7. **Justinmind.** [En línea] 2012. [Citado el: 10 de febrero de 2012.] www.justinmind.com.

8. **Brown, Dan.** *Communicating Desing: Developing Web Site Documentation and Planning.* Berkeley : New Riders, 2007.

9. **Flairbuilder.** Flairbuilder. Wireframes, Mockups, Prototypes. [En línea] Pascu C.Cristian-Constantin PFA, 2011. http://flairbuilder.com.

10. **Axure Software Solutions, Inc,2011.** [En línea] 5 de 1 de 2002. Disponible en: http://www.axure.com. http://www.axure.com/features.

11. **ForeUI.** ForeUI Easy to use UI Prototyping tool. [En línea] 2011. http://www.foreui.com/.

12. **MockFlow.** MockFlow. [En línea] A Produle Systems, 2012. www.mockflow.com.

13. **Extreme Planner.** ExtremePlanner. [En línea] Extreme Planner Software, 2012. www.extremeplanner.com/easyprototype.

14. **Pidoco.** Pidoco. [En línea] Pidoco GmbH, 2012. https://pidoco.com/en.

15. **Montoro, Mario Pérez y Codina, Lluis.** *Software de prototipado para la arquitectura de la información: funcionalidad y evaluación.* 2010.

16. **G. Peñalver, A. Meneses, S. García.** *SPX, Metodología ágil para el desarrollo de software.* La Habana: Universidad de las Ciencias Informáticas : s.n., 2010.

17. **Ágil.** *Metodología ágil.* 2010. Disponible en: http://www.seccperu.org/files/Metodologias%20Agiles.pdf.

18. **García, Felipe U. Pérez.** [En línea] 2008. http://www.larevistainformatica.com/tipo-lenguaje-programación.htm.

19. **Zukowski, John.** 2003. Disponible en: http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29.

20. **Celis, Ismael.** [En línea] 2005. http://www.webtaller.com/maletin/articulos/el-ataque-de-los-frameworks.php.

21. **Marinilli, Mauro.** Swing and SWT: A Tale of Two Java GUI Libraries. [En línea] 2006. http://www.developer.com/java/other/article.php/2179061/Swing-and-SWT-A-Tale-of-Java-GUI-Libraries.htm..

22. **Netbeans.** 2005. Disponible en: http://netbeans.org/community/releases/69/index_es.html.

23. **EcuRed.** EcuRed: Enciclopedia cubana. [En línea] [Citado el: 5 de abril de 2012.] www.ecured.cu.

24. **Visual Paradigm.** Visual Paradigm for UML 7.4. [En línea] Visual Paradigm, 2010. http://images.visual-paradigm.com/datasheet.pdf..

25. **CENIA_GAI_OT_Estándar_de_Código.2010.** Estándar de código. [En línea] https://repositorio.cenia.prod.uci.cu/svn/documnetacion/Dpto_UD/ABAD/EP_ABAD/4.general/CENIA_GAI_OT_Estándar_de_Código.pdf.

26. **Notación y estilo en programación.** britiding.com. [En línea] http://www.bitriding.com/articulos/notacion-estilo-programacion.html.

27. **Mercer, Dave.** Biblioteca.uci.cu. *Fundamentos de Programación en XML.* [En línea] 2001. http://bibliodoc.uci.cu/pdf/reg01311.pdf.958-41-0297-4.

28. **Larman, Craig.** *UML y Patrones.* s.l : Patience Hall, 1999. 970-17-0261-1.

29. **Flower, Martin.** *Patterns of Enterprise Application Architecture.* sl : Addison Wesley, 2003.

30. **Jacobson, Ivar, Boch, Grady y Rumbaugh, James.** *El proceso Unificado de Desarrollo del Software .* Madrid : Addison Wesley, 2000.

31. **Pressman, Roger.** [En línea] 1998. http://www.buenastareas.com/ensayos/Estrategias-De-Prueba-De Software-Convencionales/1045971.

32. **Pressman, Roger S.** *Ingienería de Software. Un enfoque práctico.* La Habana : Félix Varela, 2005.

Bibliografía consultada

1. **Gutiérrez, Mario Pérez-Montoro.** *Arquitectura de la información en entornos web.* Gijón : Trea, 2010. (En prensa).

2. **Sanchez de Bustamante, Antonio Montes de Oca.** *A.Arquitectura de información y usabilidad: nociones básicas para los profesionales de la información. 2004.* Ciudad Habana : Acimed, 2004. Disponible en: http://bvs.sld.cu/revistas/aci/vol12_6_04/aci04604.htm.

3. **Garret, J.J.** A visual vocabulary for describing information architecture and interaction desing. [En línea] En línea. [Citado el: 16 de febrero de 2011.] http://www.jjg.net/ia/visvocab.

4. **León, Rodrigo Ronda.** [En línea] 2007. http://www.nosolousabilidad.com/articulos/tecnicas_ai.htm.

5. **Perez, Meritxell Ramon.** *Evaluación de herramientas para prototipado de sistemas interactivos.* Lleida : s.n., 2010.

6. **Granollers, Tony.** *Una metodología que integra la Ingeniería del Software, la interacción persona-ordenador y la accesibilidad en el contexto de los equipos de desarrollo multidisciplinares.* Lleida : Univertitat de Lleida, 2004.

7. **Justinmind.** [En línea] 2012. [Citado el: 10 de febrero de 2012.] www.justinmind.com.

8. **Brown, Dan.** *Communicating Desing: Developing Web Site Documentation and Planning.* Berkeley : New Riders, 2007.

9. **Flairbuilder.** Flairbuilder. Wireframes, Mockups, Prototypes. [En línea] Pascu C.Cristian-Constantin PFA, 2011. http://flairbuilder.com.

10. **Axure Software Solutions, Inc,2011.** [En línea] 5 de 1 de 2002. Disponible en: http://www.axure.com. http://www.axure.com/features.

11. **ForeUI.** ForeUI Easy to use UI Prototyping tool. [En línea] 2011. http://www.foreui.com/.

12. **MockFlow.** MockFlow. [En línea] A Produle Systems, 2012. www.mockflow.com.

13. **Extreme Planner.** ExtremePlanner. [En línea] Extreme Planner Software, 2012. www.extremeplanner.com/easyprototype.

14. **Pidoco.** Pidoco. [En línea] Pidoco GmbH, 2012. https://pidoco.com/en.

15. **Montoro, Mario Pérez y Codina, Lluis.** *Software de prototipado para la arquitectura de la información: funcionalidad y evaluación.* 2010.

16. **G. Peñalver, A. Meneses, S. García.** *SPX, Metodología ágil para el desarrollo de software.* La Habana: Universidad de las Ciencias Informáticas : s.n., 2010.

17. **Ágil.** *Metodología ágil.* 2010. Disponible en: http://www.seccperu.org/files/Metodologias%20Agiles.pdf.

18. **García, Felipe U. Pérez.** [En línea] 2008. http://www.larevistainformatica.com/tipo-lenguaje-programación.htm.

19. **Zukowski, John.** 2003. Disponible en: http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29.

20. **Celis, Ismael.** [En línea] 2005. http://www.webtaller.com/maletin/articulos/el-ataque-de-los-frameworks.php.

21. **Marinilli, Mauro.** Swing and SWT: A Tale of Two Java GUI Libraries. [En línea] 2006. http://www.developer.com/java/other/article.php/2179061/Swing-and-SWT-A-Tale-of-Java-GUI-Libraries.htm..

22. **Netbeans.** 2005. Disponible en: http://netbeans.org/community/releases/69/index_es.html.

23. **EcuRed.** EcuRed: Enciclopedia cubana. [En línea] [Citado el: 5 de abril de 2012.] www.ecured.cu.

24. **Visual Paradigm.** Visual Paradigm for UML 7.4. [En línea] Visual Paradigm, 2010. http://images.visual-paradigm.com/datasheet.pdf..

25. **CENIA_GAI_OT_Estándar_de_Código.2010.** Estándar de código. [En línea] https://repositorio.cenia.prod.uci.cu/svn/documnetacion/Dpto_UD/ABAD/EP_ABAD/4.general/CENIA_GAI_OT_Estándar_de_Código.pdf.

26. **Notación y estilo en programación.** britiding.com. [En línea] http://www.bitriding.com/articulos/notacion-estilo-programacion.html.

27. **Mercer, Dave.** Biblioteca.uci.cu. *Fundamentos de Programación en XML.* [En línea] 2001. http://bibliodoc.uci.cu/pdf/reg01311.pdf.958-41-0297-4.

28. **Larman, Craig.** *UML y Patrones.* s.l : Patience Hall, 1999. 970-17-0261-1.

29. **Flower, Martin.** *Patterns of Enterprise Application Architecture.* sl : Addison Wesley, 2003.

30. **Jacobson, Ivar, Boch, Grady y Rumbaugh, James.** *El proceso Unificado de Desarrollo del Software .* Madrid : Addison Wesley, 2000.

31. **Pressman, Roger.** [En línea] 1998. http://www.buenastareas.com/ensayos/Estrategias-De-Prueba-De Software-Convencionales/1045971.

32. **Pressman, Roger S.** *Ingienería de Software. Un enfoque práctico.* La Habana : Félix Varela, 2005.

33. **Grupo de Ingeniería del Software y Sistemas de Información.** Ingeniería del Software y Sistemas de Información. [En línea] 2003. http://issi.dsci.upv.es/publications/archives/f-1069167248521/actas.pdf.

34. **Instituto Tecnológico de Hermosillo.** *El lenguaje Java.* [En línea] 2009. http://eddi.ith.mx/Curso/Contenido/java.htm.

35. **UML y Patrones.** *Introducción al análisis y diseño orientado a objetos.* 2011.

36. **Sun Microsystems.** Sun Developer Network. [En línea] 2010. http.//java.sun.com/javase/.

37. **2009., JAVA. 2009. JAVA. [En línea].** Java. [En línea] http://java.com/es/about/.

38. **Lago, Ramiro.** *Patrones de diseño de software.* 2007. Disponible en: http://www.proactiva-calidad.com/java/patrones/index.html.

39. **León, Rodrigo Ronda y Rábade, Yaima Mesa.** [En línea] 2005. http://www.nosolousabilidad.com/articulos/analisis_secuencia.htm.

40. **León, Rodrigo Ronda.** Arquitectura de Información: análisis histórico-conceptual. *No solo usabilidad.* [En línea] 28 de abril de 2008. [Citado el: 8 de enero de 2011.] Disponible en: http://www.nosolousabilidad.com/articulos/historia_arquitectura_informacion.htm. ISSN 1886-8592.

41. **Morville, Louis Rosenfeld Peter.** *Information Architecture for the World Wide Web, Third Edition.* s.l : O´Reilly Media, 2006.

42. **Romero, Peñalver.** *Metodología ágil para proyectos de software libre.* 2008.

43. **Teo, Javier Moldes.** *Java 2 J2se 1.4 Guías Prácticas.* s.l : Anaya Multimedia, 2003.

44. **Valls, Ignasi Pérez.** *Orígenes de los patrones.* 5 de 1 de 2009. Disponible en: http://www.javadabbadoo.org/cursos/infosintesis.net/javase/paqawt/selectorcolores/paso03patrones.html.

45. **Wells, Don.** Extreme Programming: A gentle introduction. [En línea] 2009. http://www.extremeprogramming.org/rules.html..

46. **Wurman, Richard Saul.** *Information Architecture.* Los Ángeles : Watson-Guptill Pubis, 1997.

Anexo 1

DCP Ejecutar las propiedades del componente Radiobutton

SC 3.1 Cortar componente

Escenario	Descripción	Respuesta del sistema	Flujo central
Cortar componente satisfactoriamente	La herramienta permite cortar un componente.	La herramienta permite cortar un componente de la página de trabajo.	1. Clic en el ícono de "Cortar".

SC 3.2 Copiar Componente

Escenario	Descripción	Respuesta del sistema	Flujo central
Copiar componente satisfactoriamente	El sistema permite copiar un componente.	La herramienta permite copiar un componente de la página de trabajo.	1.Clic en el menú Archivo 2. Seleccionar la opción "Nuevo proyecto".

SC 3.4 Pegar componente

Escenario	Descripción	Respuesta del sistema	Flujo central
Pegar componente correctamente.	La herramienta permite pegar un componente correctamente.	El sistema pega correctamente el componente en la página de trabajo.	1. Copiar el componente 2. Clic en el ícono "Pegar".

Pegar componente incorrectamente.	Permite pegar un componente en una página.	Muestra un mensaje “No hay ningún componente copiado”.	1.Oprimir el botón “Pegar” 2.Mostrar un mensaje ¨No hay ningún componente copiado¨ 3. Oprimir el botón “Aceptar”.

SC 3.5 Eliminar componente

Escenario	Descripción	Respuesta del sistema	Flujo central
Eliminar componente satisfactoriamente	La herramienta permite eliminar correctamente el componente	La herramienta elimina el componente seleccionado.	1. Clic en el botón “Eliminar”.

SC 3.6 Centrar vertical

Escenario	Descripción	Respuesta del sistema	Flujo central
Centrar verticalmente	La herramienta permite	Centra verticalmente el	1. Seleccionar la opción

un componente satisfactoriamente	centrar verticalmente un componente.	componente de la página de trabajo.	de “Centrar verticalmente”.

SC 3.7 Centrar horizontal

Escenario	Descripción	Respuesta del sistema	Flujo central
Central horizontalmente un componente satisfactoriamente.	La herramienta permite centrar verticalmente un componente.	Centra horizontalmente el componente de la página de trabajo.	1. Seleccionar la opción de “Centrar Verticalmente”.

SC 3.8 Ancho y alto del componente

Escenario	Descripción	Ancho	Alto	Respuesta del sistema	Flujo central
Darle el ancho y alto al componente correctamente.	La herramienta permite darle el ancho y alto que se desee a los componentes.	V (50)	V (60)	Se modifica el componente en el área de trabajo.	1.Se selecciona el componente 2. Se inserta los valores del ancho y alto del componente correctamente. 3. Se oprime el botón “Ok”.
Campos vacíos.	El usuario deja campos vacíos.	V (80)	I (vacío)	No se modifica el componente y se muestra un mensaje “Los campos no	1.Se selecciona el componente 2. Se inserta letras en el valor del ancho y alto del componente.
		I (vacío)	V (95)		
		I (vacío)	I (vacío)		

				pueden estar vacíos".	3. Se oprime el botón "Ok". 4. Se muestra un mensaje 5. Se oprime el botón "Aceptar".
Darle ancho y alto a un componente con datos incorrectos	El usuario introduce datos incorrectos en la variable.	I (pepe)	I (Pepe)	No se modifica el alto y el ancho y se muestra un mensaje ¨Introduzca los datos correctamente¨	1.Se selecciona el componente 2. Se inserta letras en el valor del ancho y alto del componente. 3. Se oprime el botón "Ok". 4. Se muestra un mensaje ¨Introduzca los datos correctamente.¨ 5. Se introducen los datos 6. Se oprime el botón "Aceptar".

SC 3.9 Editar texto

Escenario	Descripción	Texto	Respuesta del sistema	Flujo central
Editar el texto del componente correctamente.	La herramienta permite escribir el texto que se desee a los componentes.	V (Opción1)	Se modifica el componente en el área de trabajo.	1.Se selecciona el componente 2. Se inserta el texto del componente correctamente. 3. Se oprime el botón "Ok".
Campos vacíos	El usuario deja campos en blanco.	V (80) I (vacío) I (vacío)	No se modifica el componente y se muestra un mensaje "El campo no puede estar vacío".	1.Se selecciona el componente 2. Se dejan campos en blanco. 3. Se oprime el botón "Ok". 4. Se muestra un mensaje 5. Se oprime el botón "Aceptar".

SC 3.10 Mover arriba

Escenario	Descripción	Respuesta del sistema	Flujo central

Mover arriba un componente satisfactoriamente.	La herramienta permite mover hacia arriba a un componente.	Mueve el componente hacia arriba del área de trabajo.	1. En la ventana de las propiedades de los componentes seleccionar la opción de “Mover arriba”.

SC 3.11 Mover abajo

Escenario	Descripción	Respuesta del sistema	Flujo central
Mover abajo a un componente satisfactoriamente.	La herramienta permite mover hacia abajo a un componente.	Mueve hacia abajo el componente en el área de trabajo.	1. En la ventana de las propiedades de los componentes seleccionar la opción de “Mover abajo”.

SC 3.12 Mover derecha

Escenario	Descripción	Respuesta del sistema	Flujo central
Mover derecha a un componente satisfactoriamente.	La herramienta permite mover hacia derecha a un componente.	Mueve a la derecha el componente en el área de trabajo.	1. En la ventana de las propiedades de los componentes seleccionar la opción de “Mover a la derecha”.

SC 3.13 Mover izquierda

Escenario	Descripción	Respuesta del sistema	Flujo central
Mover Izquierda a un componente	La herramienta permite mover hacia Izquierda a	Mueve a la izquierda el componente en el área	1. En la ventana de las propiedades de los

satisfactoriamente	un componente.	de trabajo.	componentes seleccionar la opción de “Mover a la izquierda”.

SC 3.14 Visible

Escenario	Descripción	Respuesta del sistema	Flujo central
No marcar la opción de visible del componente	La herramienta permite ver o no ver el componente en el área de trabajo.	El componente no se ve en el área de trabajo	1. En la ventana de las propiedades de los componentes seleccionar la opción de “Visible”.
Marcar la opción de visible del componente		El componente se ve en el área de trabajo.	

SC 3.15 Color de texto

Escenario	Descripción	Color	Respuesta del sistema	Flujo central
Darle un color al texto del componente con el botón aceptar.	Se selecciona un color y se modifica el color del texto del componente.	V (azul)	Se pinta el texto del componente del color azul	1.En la ventana de las propiedades de los componentes 2. Seleccionar la opción de color de texto. 3.Se muestra una ventana 4. Seleccionar el color. 5. Oprimir el botón aceptar.
		V (azul)	No se pinta el	1.En la ventana de las

Darle un color al texto del componente.			componente	propiedades de los componentes 2. Seleccionar la opción de color de texto. 3.Se muestra una ventana 4. Seleccionar el color. 5. Oprimir el botón Cancelar
Darle color al texto del componente.			El componente restablece el color que tenía inicialmente.	1.En la ventana de las propiedades de los componentes 2. Seleccionar la opción de color de texto. 3.Se muestra una ventana 4. Seleccionar el color. 5. Oprimir el botón Restablecer

SC 3.16 Tipo de letra

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
Editar el tipo de letra del componente satisfactoriamente.	La herramienta permite editar el tipo de letra del componente.	Edita el tipo de letra del componente.	1.En la ventana de las propiedades de los componentes 2. Seleccionar la opción “Fuente”. 3. Muestra la ventana para seleccionar el tipo

			de letra 4. Oprime el botón “Aceptar”.

Descripción de las variables

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1.	Ancho	Entero	No	No pueden ser letras ni caracteres.
2.	Alto	Entero	No	No pueden ser letras ni caracteres.
3	texto	String	No	No puede ser caracteres solos